

GRUNDLAGEN SEMANTIC WEB

Lehrveranstaltung im WS10/11
Seminar für Computerlinguistik
Universität Heidelberg

Dr. Sebastian Rudolph
Institut AIFB
Universität Karlsruhe

OWL - SYNTAX & INTUITION

Dr. Sebastian Rudolph

Einleitung und Ausblick

XML und URIs

Einführung in RDF

RDF Schema

Logik - Grundlagen

Semantik von RDF(S)

OWL - Syntax und Intuition

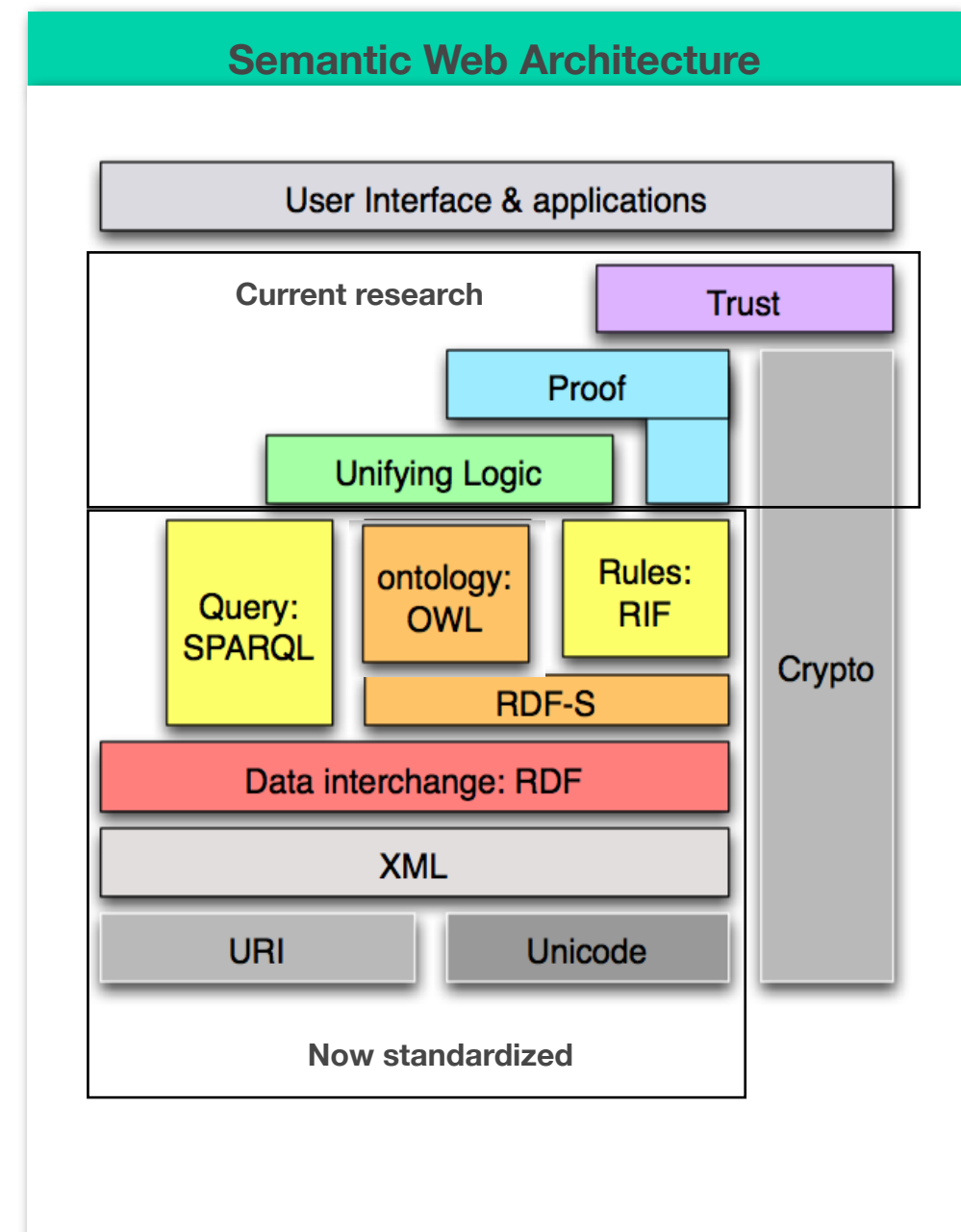
OWL - Semantik und Reasoning

SPARQL - Syntax und Intuition

Semantik von SPARQL und konjunktive Anfragen

OWL 1.1 - Syntax und Semantik

Semantic Web und Regeln



OWL - SYNTAX & INTUITION

Dr. Sebastian Rudolph

Einleitung und Ausblick

XML und URIs

Einführung in RDF

RDF Schema

Logik - Grundlagen

Semantik von RDF(S)

OWL - Syntax und Intuition

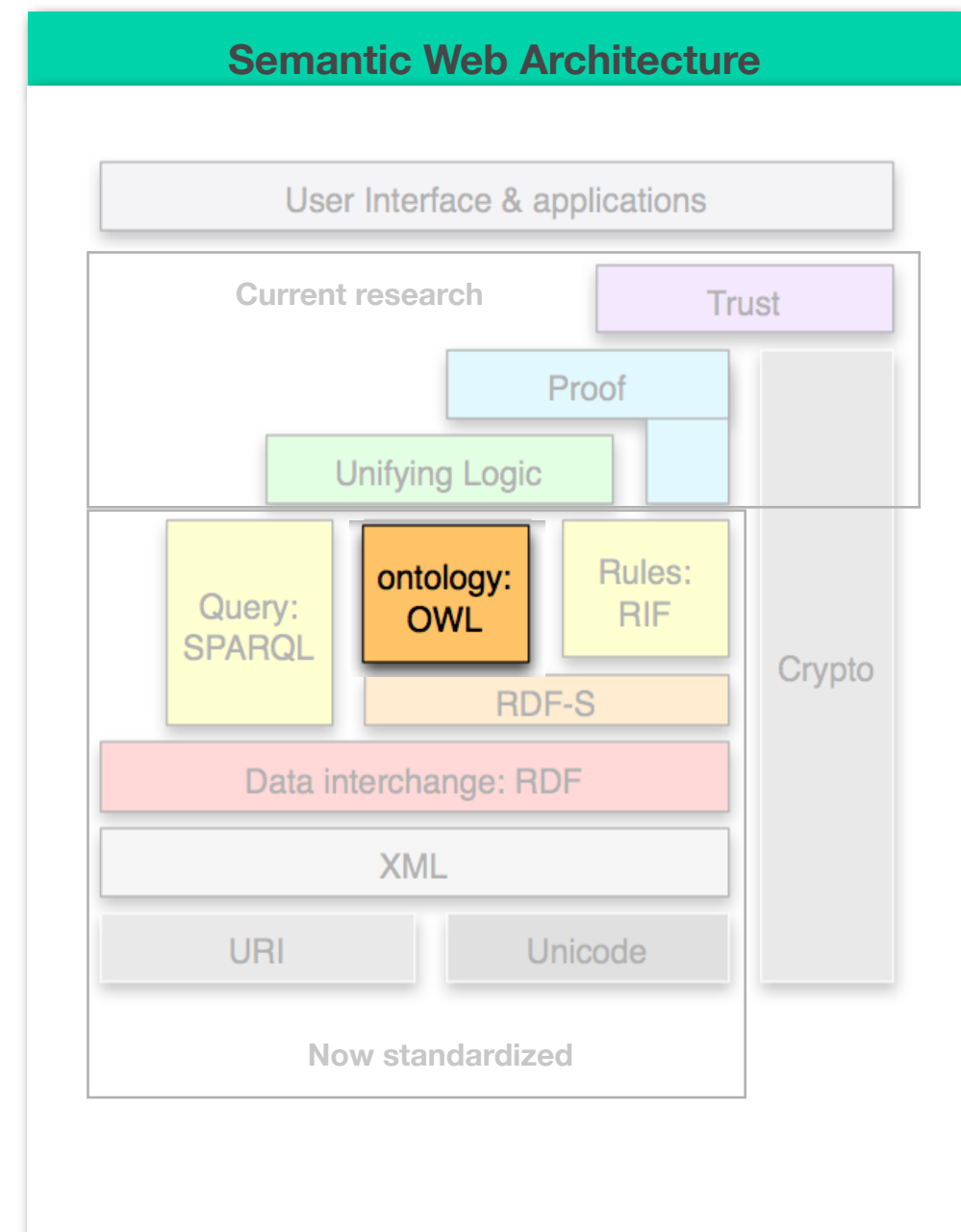
OWL - Semantik und Reasoning

SPARQL - Syntax und Intuition

Semantik von SPARQL und konjunktive Anfragen

OWL 1.1 - Syntax und Semantik

Semantic Web und Regeln



AGENDA



- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- Anfragen an OWL-Ontologien

AGENDA

- **Motivation**
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- Anfragen an OWL-Ontologien

- Begriff existiert nur in der Einzahl (es gibt also keine „Ontologien“)
- bezeichnet die „*Lehre vom Sein*“
- zu finden bei Aristoteles (Sokrates), Thomas von Aquin, Descartes, Kant, Hegel, Wittgenstein, Heidegger, Quine, ...

Gruber (1993):

„An Ontology is a
formal specification
of a shared
conceptualization
of a domain of interest“

Gruber (1993):

„An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest“

⇒ maschinell interpretierbar

Gruber (1993):

„An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest“

⇒ maschinell interpretierbar

⇒ beruht auf Konsens

Gruber (1993):

„An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest“

⇒ maschinell interpretierbar

⇒ beruht auf Konsens

⇒ beschreibt Begrifflichkeiten

Gruber (1993):

„An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest“

⇒ maschinell interpretierbar

⇒ beruht auf Konsens

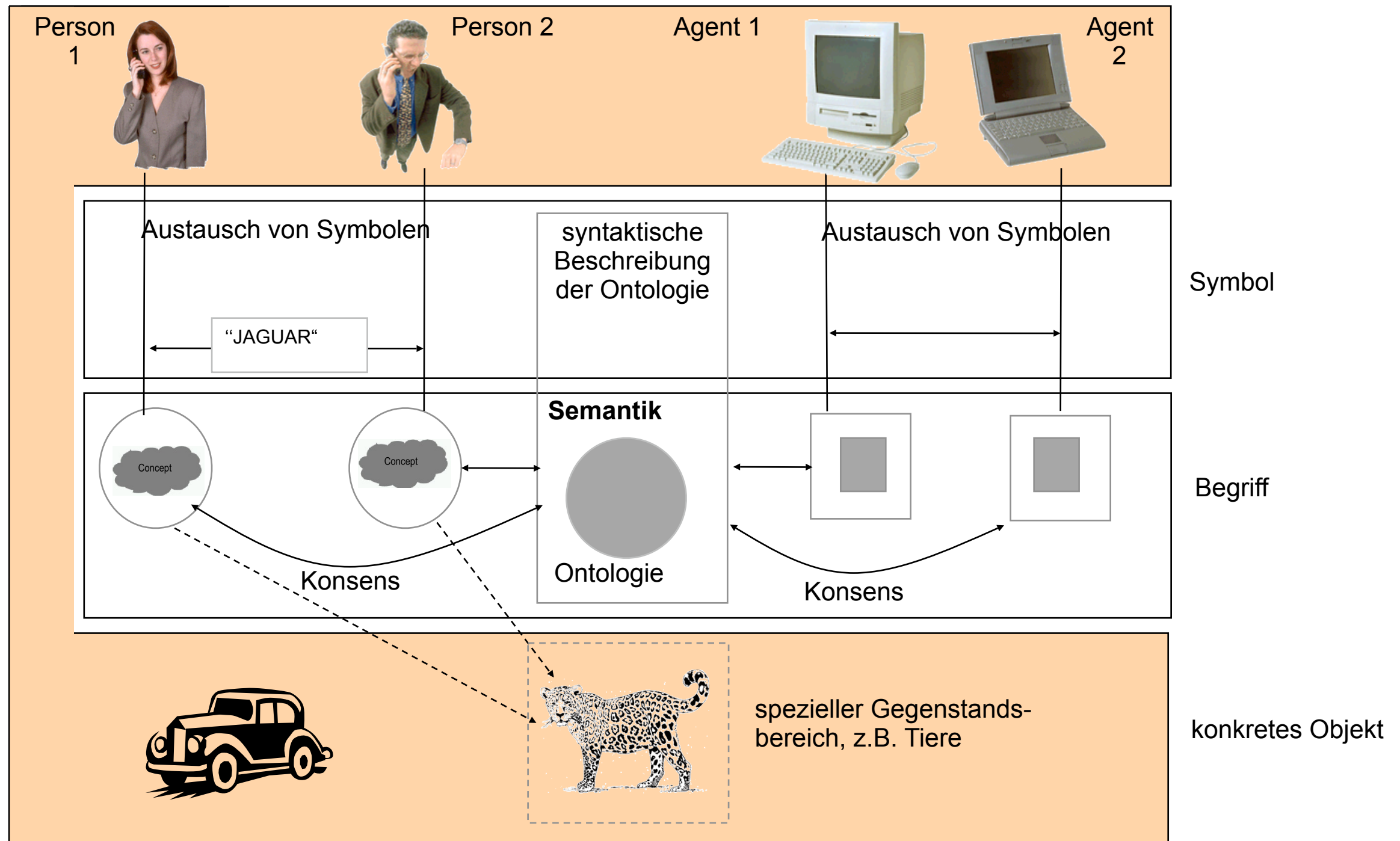
⇒ beschreibt Begrifflichkeiten

⇒ bezogen auf ein „Thema“
(Gegenstandsbereich)

Hitzler, Krötzsch, Rudolph (2009):

"In computer science, an ontology is a description of knowledge about a domain of interest, the core of which is a machine-processable specification with a formally defined meaning."

ONTOLOGIE & KOMMUNIKATION



RDF SCHEMA ALS ONTOLOGIESPRACHE?

AIFB 

- geeignet für einfache Ontologien
- Vorteil: automatisches Schlussfolgern ist relativ effizient
- aber: für komplexere Modellierungen ungeeignet
- Rückgriff auf mächtigere Sprachen, wie
 - OWL
 - F-Logik

AGENDA

AIFB 

- Motivation
- **OWL - Allgemeines**
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- Anfragen an OWL-Ontologien

- W3C Recommendation seit 2004
- Semantisches Fragment von FOL
- Drei Varianten:
 $\text{OWL Lite} \subseteq \text{OWL DL} \subseteq \text{OWL Full}$
- Keine Reifikation in OWL DL
→ RDFS ist Fragment von OWL Full
- OWL DL ist entscheidbar
entspricht der Beschreibungslogik *SHOIN*(D)
- W3C-Dokumente (Vorlesungswebseite) enthalten Details,
die hier nicht alle angesprochen werden können.

OWL DOKUMENTE

AIFB 

- sind RDF Dokumente
(zumindest in der Standard-Syntax; es gibt auch andere)
- bestehen aus
 - Kopf mit allgemeinen Angaben
 - Rest mit der eigentlichen Ontologie

- Definition von Namespaces in der Wurzel

```
<rdf:RDF
```

```
  xmlns    ="http://www.semanticweb-grundlagen.de/beispielontologie#"
```

```
  xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
  xmlns:xsd  ="http://www.w3.org/2001/XMLSchema#"
```

```
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
```

```
  xmlns:owl  ="http://www.w3.org/2002/07/owl#">
```

```
...
```

```
</rdf:RDF>
```

- Allgemeine Informationen

```
<owl:Ontology rdf:about="">
```

```
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/
XMLSchema#string">
```

```
    SWRC Ontologie in der Version vom Dezember 2005
```

```
  </rdfs:comment>
```

```
  <owl:versionInfo>v0.5</owl:versionInfo>
```

```
  <owl:imports rdf:resource="http://www.semanticweb-
    grundlagen.de/foo"/>
```

```
  <owl:priorVersion      rdf:resource="http://ontoware.org/
    projects/swrc"/>
```

```
</owl:Ontology>
```

DER KOPF EINES OWL DOKUMENTES



von RDFS geerbt

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`

außerdem

- `owl:imports`

für Versionierung

- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`

AGENDA

AIFB 

- Motivation
- OWL - Allgemeines
- **Klassen, Rollen und Individuen**
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- Anfragen an OWL-Ontologien

KLASSEN, ROLLEN UND INDIVIDUEN



- Die drei Bausteine von Ontologieaxiomen.
- Klassen
 - Vergleichbar mit Klassen in RDFS
- Individuen
 - Vergleichbar mit Objekten in RDFS
- Rollen
 - Vergleichbar mit Property's in RDFS

- Definition

```
<owl:Class rdf:ID="Professor"/>
```

- vordefiniert:

- owl:Thing
- owl:Nothing

- Definition durch Klassenzugehörigkeit

```
<rdf:Description rdf:ID="RudiStuder">  
<rdf:type rdf:resource="#Professor"/>  
</rdf:Description>
```

- gleichbedeutend:

```
<Professor rdf:ID="RudiStuder"/>
```


ABSTRAKTE ROLLEN



- abstrakte Rollen werden definiert wie Klassen

```
<owl:ObjectProperty  
  rdf:ID="Zugehoerigkeit"/>
```

- Domain und Range abstrakter Rollen

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">  
  <rdfs:domain rdf:resource="#Person"/>  
  <rdfs:range rdf:resource="#Organisation"/>  
</owl:ObjectProperty>
```

KONKRETE ROLLEN



- konkrete Rollen haben Datentypen im Range
`<owl:DatatypeProperty rdf:ID="Vorname"/>`
- Domain und Range konkreter Rollen
`<owl:DatatypeProperty rdf:ID="Vorname">`
`<rdfs:domain rdf:resource="#Person" />`
`<rdfs:range rdf:resource="&xsd:string"/>`
`</owl:DatatypeProperty>`
- Viele XML Datentypen können verwendet werden.
Im Standard vorgeschrieben sind `integer` und `string`.

INDIVIDUEN UND ROLLEN



```
<Person rdf:ID="RudiStuder">
```

```
  <Zugehoerigkeit rdf:resource="#AIFB"/>
```

```
  <Zugehoerigkeit rdf:resource="#ontoprise"/>
```

```
  <Vorname rdf:datatype="&xsd:string">Rudi</Vorname>
```

```
</Person>
```

- Rollen sind im allgemeinen nicht funktional.

AGENDA



- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- **Klassenbeziehungen**
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- Anfragen an OWL-Ontologien

EINFACHE KLASSENBEZIEHUNGEN



```
<owl:Class rdf:ID="Professor">  
  <rdfs:subClassOf  
    rdf:resource="#Fakultaetsmitglied"/>  
</owl:Class>  
  
<owl:Class rdf:ID="Fakultaetsmitglied">  
  <rdfs:subClassOf rdf:resource="#Person"/>  
</owl:Class>
```

Es folgt durch Inferenz, dass `Professor` eine Subklasse von `Person` ist.

EINFACHE KLASSENBEZIEHUNGEN

AIFB 

```
<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf
    rdf:resource="#Fakultaetsmitglied"/>
</owl:Class>

<owl:Class rdf:ID="Buch">
  <rdfs:subClassOf rdf:resource="#Publikation"/>
</owl:Class>

<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Es folgt durch Inferenz, dass Professor und Buch ebenfalls disjunkte Klassen sind.

EINFACHE KLASSENBEZIEHUNGEN



```
<owl:Class rdf:ID="Buch">  
  <rdfs:subClassOf rdf:resource="#Publikation"/>  
</owl:Class>
```

```
<owl:Class rdf:about="#Publikation">  
  <owl:equivalentClass  
    rdf:resource="#Publikation"/>  
</owl:Class>
```

Es folgt durch Inferenz, dass Buch eine Subklasse von Publication ist.

INDIVIDUEN UND KLASSENBEZIEHUNGEN



```
<Buch rdf:ID="SemanticWebGrundlagen">
```

```
  <Autor rdf:resource="#PascalHitzler"/>
```

```
  <Autor rdf:resource="#MarkusKrötzsch"/>
```

```
  <Autor rdf:resource="#SebastianRudolph"/>
```

```
  <Autor rdf:resource="#YorkSure"/>
```

```
</Buch>
```

```
<owl:Class rdf:about="#Buch">
```

```
  <rdfs:subClassOf rdf:resource="#Publikation"/>
```

```
</owl:Class>
```

**Es folgt durch Inferenz, dass
SemanticWebGrundlagen eine
Publikation ist.**

BEZIEHUNGEN ZWISCHEN INDIVIDUEN

AIFB 

```
<Professor rdf:ID="RudiStuder"/>  
  
<rdf:Description rdf:about="#RudiStuder">  
  <owl:sameAs  
    rdf:resource="#ProfessorStuder"/>  
</rdf:Description>
```

Es folgt durch Inferenz, dass `ProfessorStuder` ein
`Professor` ist.

Verschiedenheit von Individuen mittels

`owl:differentFrom`.

BEZIEHUNGEN ZWISCHEN INDIVIDUEN



```
<owl:AllDifferent>
```

```
<owl:distinctMembers  
  rdf:parseType="Collection">
```

```
<Person rdf:about="#RudiStuder"/>
```

```
<Person rdf:about="#YorkSure"/>
```

```
<Person rdf:about="#PascalHitzler"/>
```

```
</owl:distinctMembers>
```

```
</owl:AllDifferent>
```

Abgekürzte Schreibweise anstelle der Verwendung von mehreren
`owl:differentFrom`.

Der Einsatz von `owl:AllDifferent` und `owl:distinctMembers`
ist nur dafür vorgesehen.

ABGESCHLOSSENE KLASSEN

AIFB 

```
<owl:Class rdf:about=#SekretaerinnenVonStuder>  
  <owl:oneOf rdf:parseType="Collection">  
    <Person rdf:about="#GiselaSchillinger"/>  
    <Person rdf:about="#SusanneWinter"/>  
  </owl:oneOf>  
</owl:Class>
```

Dies besagt, dass es nur **genau diese beiden** SekretarinnenVonStuder gibt.

AGENDA



- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- **komplexe Klassen**
- Eigenschaften von Rollen
- OWL Varianten
- Anfragen an OWL-Ontologien

LOGISCHE KLASSENKONSTRUKTOREN



- logisches Und (Konjunktion):
`owl:intersectionOf`
- logisches Oder (Disjunktion):
`owl:unionOf`
- logisches Nicht (Negation):
`owl:complementOf`
- Werden verwendet, um komplexe Klassen aus einfachen Klassen zu konstruieren.

KONJUNKTION

AIFB 

```
<owl:Class rdf:about="#SekretaerinnenVonStuder">  
  <owl:equivalentClass>  
    <owl:intersectionOf  
      rdf:parseType="Collection">  
        <owl:Class rdf:about="#Sekretaerinnen"/>  
        <owl:Class  
          rdf:about="#AngehoeerigeAGStuder"/>  
      </owl:intersectionOf>  
    </owl:equivalentClass>  
  </owl:Class>
```

Es folgt z.B. durch Inferenz, dass alle
SekretaerinnenVonStuder **auch**
Sekretaerinnen **sind**.

DISJUNKTION



```
<owl:Class rdf:about="#Professor">  
  <owl:subClassOf>  
    <owl:unionOf rdf:parseType="Collection">  
      <owl:Class rdf:about="#aktivLehrend"/>  
      <owl:Class rdf:about="#imRuhestand"/>  
    </owl:unionOf>  
  </owl:subClassOf>  
</owl:Class>
```

NEGATION



```
<owl:Class rdf:about="#Fakultaetsmitglied">  
  <owl:subClassOf>  
    <owl:complementOf rdf:resource="#Publikation"/>  
  </owl:subClassOf>  
</owl:Class>
```

semantisch äquivalente Aussage:

```
<owl:Class rdf:about="#Fakultaetsmitglied">  
  <owl:disjointWith rdf:resource="#Publikation"/>  
</owl:Class>
```


ROLLENEINSCHRÄNKUNGEN (*ALL VALUES FROM*)

AIFB 

- dienen der Definition komplexer Klassen durch Rollen

```
<owl:Class rdf:ID="Pruefung">
```

```
<rdfs:subClassOf>
```

```
<owl:Restriction>
```

```
<owl:onProperty rdf:resource="#hatPruefer"/>
```

```
<owl:allValuesFrom rdf:resource="#Professor"/>
```

```
</owl:Restriction>
```

```
</rdfs:subClassOf>
```

```
</owl:Class>
```

D.h. *alle* Prüfer einer Prüfung müssen Professoren sein.

ROLLENEINSCHRÄNKUNGEN (SOME VALUES FROM)

```
<owl:Class rdf:about="#Pruefung">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#hatPruefer"/>  
      <owl:someValuesFrom rdf:resource="#Person"/>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

D.h. jede Prüfung muss *mindestens einen* Prüfer haben.

ROLLENEINSCHRÄNKUNGEN (KARDINALITÄTEN)



```
<owl:Class rdf:about="#Pruefung">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#hatPruefer"/>  
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">  
        2  
      </owl:maxcardinality>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

Eine Prüfung kann *höchstens* zwei Prüfer haben.

ROLLENEINSCHRÄNKUNGEN (KARDINALITÄTEN)



```
<owl:Class rdf:about="#Pruefung">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#hatThema"/>  
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">  
        3  
      </owl:minCardinality>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

Eine Prüfung muss sich über *mindestens drei* Themengebiete erstrecken.

ROLLENEINSCHRÄNKUNGEN (KARDINALITÄTEN)



```
<owl:Class rdf:about="#Pruefung">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#hatThema"/>  
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">  
        3  
      </owl:cardinality>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

Eine Prüfung muss sich über *genau drei* Themengebiete erstrecken.

ROLLENEINSCHRÄNKUNGEN (HASVALUE)



```
<owl:Class rdf:ID="PruefungBeiStuder">
```

```
<rdfs:equivalentClass>
```

```
<owl:Restriction>
```

```
<owl:onProperty rdf:resource="#hatPruefer"/>
```

```
<owl:hasValue rdf:resource="#RudiStuder"/>
```

```
</owl:Restriction>
```

```
</rdfs:equivalentClass>
```

```
</owl:Class>
```

`owl:hasValue` verweist immer auf eine konkrete Instanz. Dies ist äquivalent zum Beispiel auf der nächsten Folie.

ROLLENEINSCHRÄNKUNGEN (HASVALUE)



```
<owl:Class rdf:ID="PruefungBeiStuder">  
  <rdfs:equivalentClass>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#hatPruefer"/>  
      <owl:someValuesFrom>  
        <owl:oneOf rdf:parseType="Collection">  
          <owl:Thing rdf:about=#RudiStuder/>  
        </owl:oneOf>  
      </owl:someValuesFrom>  
    </owl:Restriction>  
  </rdfs:equivalentClass>  
</owl:Class>
```

AGENDA



- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- **Eigenschaften von Rollen**
- OWL Varianten
- Anfragen an OWL-Ontologien

ROLLENBEZIEHUNGEN

AIFB 

```
<owl:ObjectProperty rdf:ID="hatPruefer">  
  <rdfs:subPropertyOf  
    rdf:resource="#hatAnwesenden"/>  
</owl:ObjectProperty>
```

Ebenso: `owl:equivalentProperty`

Rollen können auch invers zueinander sein:

```
<owl:ObjectProperty rdf:ID="hatPruefer">  
  <owl:inverseOf rdf:resource="#prueferVon"/>  
</owl:ObjectProperty>
```

ROLLENEIGENSCHAFTEN



- Domain
- Range
- Transitivität, d.h.
 $r(a,b)$ und $r(b,c)$ impliziert $r(a,c)$
- Symmetrie, d.h.
 $r(a,b)$ impliziert $r(b,a)$
- Funktionalität
 $r(a,b)$ und $r(a,c)$ impliziert $b=c$
- Inverse Funktionalität
 $r(a,b)$ und $r(c,b)$ impliziert $a=c$

DOMAIN UND RANGE



```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">  
  <rdfs:range rdf:resource="#Organisation"/>  
</owl:ObjectProperty>
```

ist gleichbedeutend mit dem Folgenden:

```
<owl:Class rdf:about="\&owl;Thing">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#Zugehoerigkeit"/>  
      <owl:allValuesFrom rdf:resource="#Organisation"/>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

DOMAIN UND RANGE: VORSICHT!

AIFB 

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">  
  <rdfs:range rdf:resource="#Organisation"/>  
</owl:ObjectProperty>  
  
<Zahl rdf:ID="Fuenf">  
  <Zugehoerigkeit rdf:resource="#Primzahlen"/>  
</Zahl>
```

Es folgt nun, dass `Primzahlen` eine
Organisation ist!

ROLLENEIGENSCHAFTEN



```
<owl:ObjectProperty rdf:ID="hatKollegen">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hatProjektleiter">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="istProjektleiterFuer">
  <rdf:type
    rdf:resource="&owl;InverseFunctionalProperty"/>
</owl:ObjectProperty>
<Person rdf:ID="YorkSure">
  <hatKollegen rdf:resource="#PascalHitzler"/>
  <hatKollegen rdf:resource="#AnupriyaAnkolekar"/>
  <istProjektleiterFuer rdf:resource="#SEKT"/>
</Person>
<Projekt rdf:ID="SmartWeb">
  <hatProjektleiter rdf:resource="#PascalHitzler"/>
  <hatProjektleiter rdf:resource="#HitzlerPascal"/>
</Projekt>
```

FOLGERUNGEN AUS DEM BEISPIEL



- AnupriyaAnkolekar hatKollegen
YorkSure
- AnupriyaAnkolekar hatKollegen
PascalHitzler
- PascalHitzler owl:sameAs
HitzlerPascal

AGENDA



- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- **OWL Varianten**
- Anfragen an OWL-Ontologien

OWL VARIANTEN

AIFB 

- OWL Full
 - Enthält OWL DL und OWL Lite
 - Enthält als einzige OWL-Teilsprache ganz RDFS
 - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind.
 - Unentscheidbar.
 - Wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.
- OWL DL
 - Enthält OWL Lite und ist Teilsprache von OWL Full.
 - Entscheidbar.
 - Wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.
 - Komplexität NExpTime (worst-case).
- OWL Lite
 - Ist Teilsprache von OWL DL und OWL Full.
 - Entscheidbar.
 - Wenig ausdrucksstark.
 - Komplexität ExpTime (worst-case).

OWL Full



- Uneingeschränkte Nutzung aller OWL und RDFS-Sprachelemente (muss gültiges RDFS sein).
- Schwierig z.B.: nicht vorhandene Typentrennung (Klassen, Rollen, Individuen), dadurch:
 - `owl:Thing` **dasselbe wie** `rdfs:resource`
 - `owl:Class` **dasselbe wie** `rdfs:Class`
 - `owl:DatatypeProperty` **Subklasse von** `owl:ObjectProperty`
 - `owl:ObjectProperty` **dasselbe wie** `rdf:Property`

TYPENDURCHMISCHUNG IN OWL FULL

AIFB 

```
<owl:Class rdf:about="#Buch">  
  <englischerName rdf:datatype="&xsd:string">  
    book  
  </englischerName>  
  <franzoesischerName rdf:datatype="&xsd:string">  
    livre  
  </franzoesischerName>  
</owl:Class>
```

Inferenzen über solche Konstrukte werden oft nicht wirklich benötigt.

OWL DL



- Nur Verwendung von explizit erlaubten RDFS Sprachelementen (z.B. die in unseren Beispielen).
Nicht erlaubt: `rdfs:Class`, `rdfs:Property`
- Typentrennung. Klassen und Rollen müssen explizit deklariert werden.
- Konkrete Rollen dürfen nicht als transitiv, symmetrisch, invers oder invers funktional deklariert werden.
- Zahlenrestriktionen dürfen nicht mit transitiven Rollen, deren Subrollen, oder Inversen davon verwendet werden.

- alle Einschränkungen für OWL DL
- außerdem:
 - nicht erlaubt: `oneOf`, `unionOf`,
`complementOf`, `hasValue`, `disjointWith`
 - Zahlenrestriktionen nur mit 0 und 1 erlaubt.
 - Einige Einschränkungen zum Auftreten von
anonymen (komplexen) Klassen, z.B. nur im
Subjekt von `rdfs:subClassOf`.

AGENDA

AIFB 

- Motivation
- OWL - Allgemeines
- Klassen, Rollen und Individuen
- Klassenbeziehungen
- komplexe Klassen
- Eigenschaften von Rollen
- OWL Varianten
- **Anfragen an OWL-Ontologien**

TERMINOLOGISCHE ANFRAGEN AN OWL (NUR KLASSEN UND ROLLEN)



- Klassenäquivalenz
- Subklassenbeziehung
- Disjunktheit von Klassen
- globale Konsistenz (Erfüllbarkeit, Widerspruchsfreiheit)
- Klassenkonsistenz: Eine Klasse ist *inkonsistent*, wenn sie äquivalent zu `owl:Nothing` ist - dies deutet oft auf einen Modellierungsfehler hin:

```
<owl:Class rdf:about="#Buch">
```

```
  <owl:subClassOf rdf:resource="#Publikation"/>
```

```
  <owl:disjointWith rdf:resource="#Publikation"/>
```

```
</owl:Class>
```

ASSERTIONALE ANFRAGEN AN OWL (MIT INDIVIDUEN)



- Instanzüberprüfung: Gehört gegebenes Individuum zu gegebener Klasse?
- Suche nach allen Individuen, die in einer Klasse enthalten sind.
- Werden zwei gegebene Individuen durch Rolle verknüpft?
- Suche nach allen Individuenpaaren, die durch eine Rolle verknüpft sind.
- ...Vorsicht: es wird nur nach „beweisbaren“ Antworten gesucht!

- Editoren
 - Protégé, <http://protege.stanford.edu>
 - SWOOP, <http://www.mindswap.org/2004/SWOOP/>
 - OWL Tools, <http://owltools.ontoware.org/>
- Inferenzmaschinen
 - Pellet, <http://www.mindswap.org/2003/pellet/index.shtml>
 - KAON2, <http://kaon2.semanticweb.org>
 - FACT++, <http://owl.man.ac.uk/factplusplus/>
 - Racer, <http://www.racer-systems.com/>

OWL SPRACHELEMENTE



Kopf

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`
- `owl:imports`

Beziehungen zwischen Individuen

- `owl:sameAs`
- `owl:differentFrom`
- `owl:AllDifferent`
(zusammen mit `owl:distinctMembers`)

Vorgeschriebene Datentypen

- `xsd:string`
- `xsd:integer`

OWL SPRACHELEMENTE



Klassenkonstruktoren und -beziehungen

- `owl:Class`
- `owl:Thing`
- `owl:Nothing`
- `rdfs:subClassOf`
- `owl:disjointWith`
- `owl:equivalentClass`
- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`

Rollenrestriktionen

- `owl:allValuesFrom`
- `owl:someValuesFrom`
- `owl:hasValue`
- `owl:cardinality`
- `owl:minCardinality`
- `owl:maxCardinality`
- `owl:oneOf`

Rollenkonstruktoren, -beziehungen und -eigenschaften

- `owl:ObjectProperty`
- `owl:DatatypeProperty`
- `rdfs:subPropertyOf`
- `owl:equivalentProperty`
- `owl:inverseOf`
- `rdfs:domain`
- `rdfs:range`
- `rdf:resource="&owl;TransitiveProperty"`
- `rdf:resource="&owl;SymmetricProperty"`
- `rdf:resource="&owl;FunctionalProperty"`
- `rdf:resource="&owl;InverseFunctionalProperty"`

WEITERFÜHRENDE LITERATUR



- <http://www.w3.org/2004/OWL/>
zentrale W3C Webseite für OWL.
- <http://www.w3.org/TR/owl-features/>
Überblick über OWL.
- <http://www.w3.org/TR/owl-ref/>
vollständige Beschreibung der OWL-Sprachkomponenten.
- <http://www.w3.org/TR/owl-guide/>
zeigt, wie OWL zur Wissensmodellierung verwendet werden kann.
- <http://www.w3.org/TR/owl-semantics/>
beschreibt die Semantik von OWL, die wir auf andere Weise später behandeln werden. Es beschreibt außerdem die abstrakte Syntax für OWL DL, die wir hier später noch ansprechen.
- Deutsche Übersetzungen mancher W3C Dokumente findet man unter <http://www.w3.org/2005/11/Translations/Lists/ListLang-de.html>

OWL - SEMANTIK & REASONING

Dr. Sebastian Rudolph

Einleitung und Ausblick

XML und URIs

Einführung in RDF

RDF Schema

Logik - Grundlagen

Semantik von RDF(S)

OWL - Syntax und Intuition

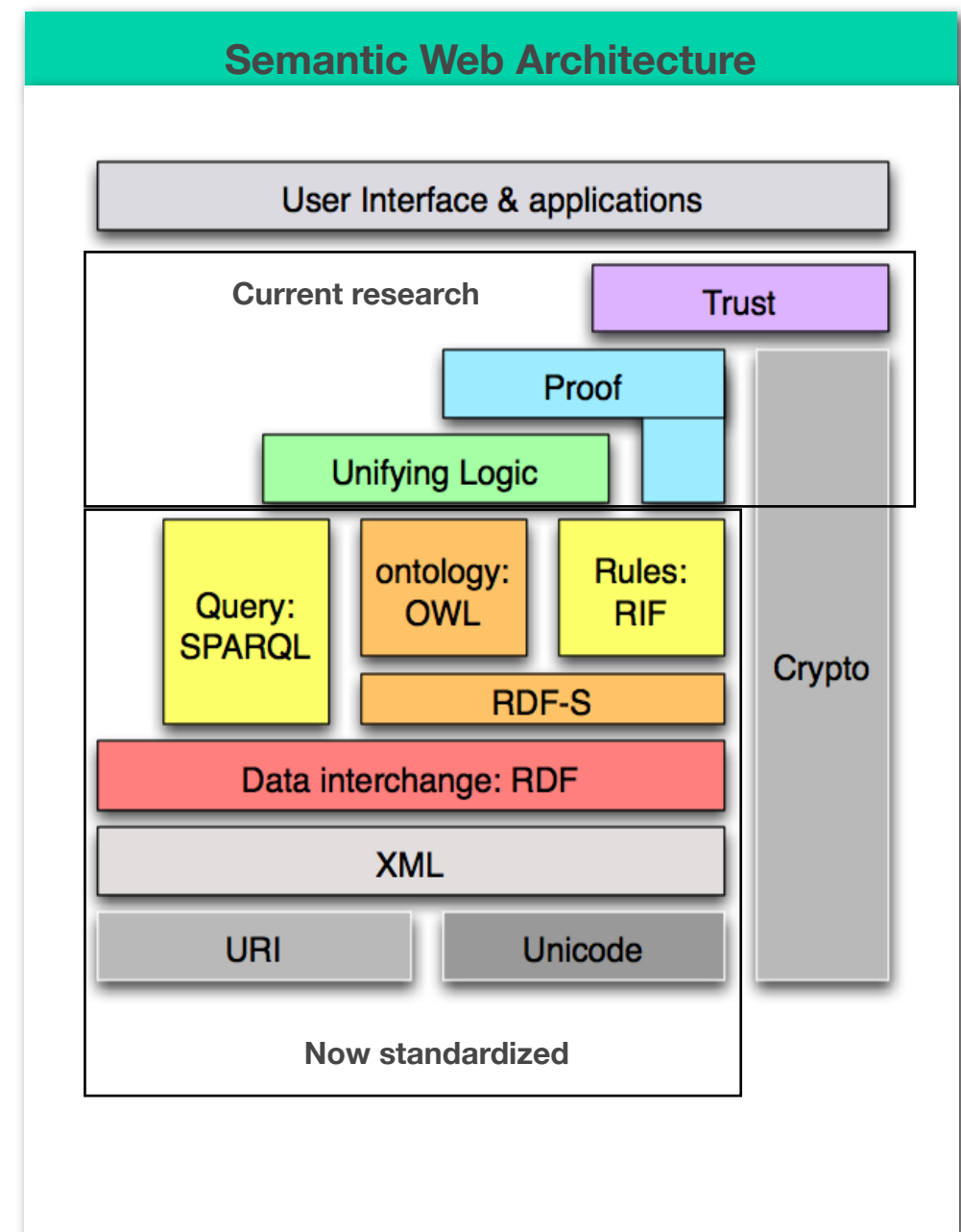
OWL - Semantik und Reasoning

SPARQL - Syntax und Intuition

Semantik von SPARQL und konjunktive Anfragen

OWL 1.1 - Syntax und Semantik

Semantic Web und Regeln



OWL - SEMANTIK & REASONING

Dr. Sebastian Rudolph

Einleitung und Ausblick

XML und URIs

Einführung in RDF

RDF Schema

Logik - Grundlagen

Semantik von RDF(S)

OWL - Syntax und Intuition

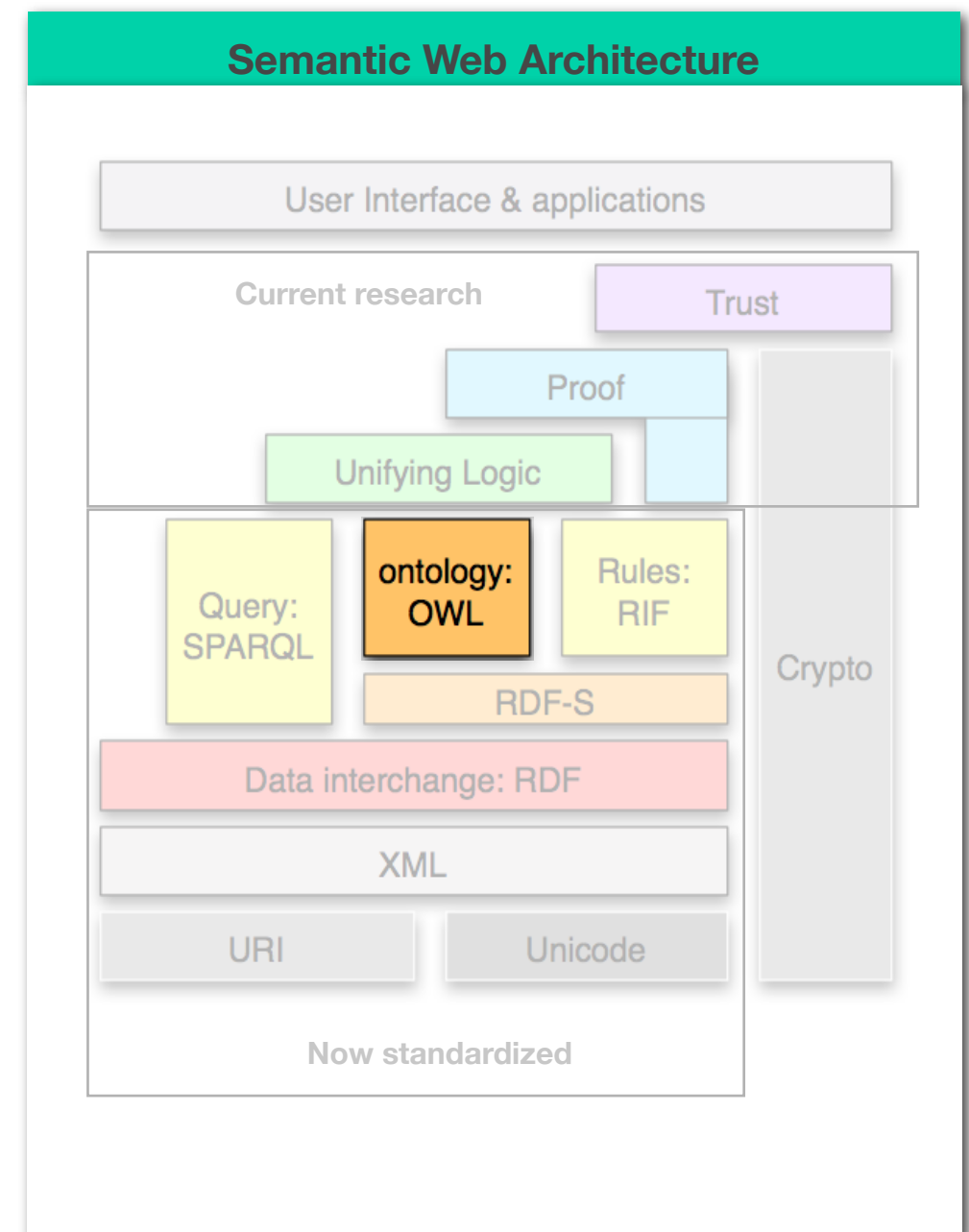
OWL - Semantik und Reasoning

SPARQL - Syntax und Intuition

Semantik von SPARQL und konjunktive Anfragen

OWL 1.1 - Syntax und Semantik

Semantic Web und Regeln



AGENDA

AIFB 

- Beschreibungslogiken
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

AGENDA

- **Beschreibungslogiken**
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

BESCHREIBUNGSLOGIKEN



- engl.: description logics (DLs)
 - Fragmente von FOL
 - meist entscheidbar
 - vergleichsweise ausdrucksstark
 - entwickelt aus semantischen Netzwerken
 - enge Verwandtschaft mit Modallogiken
-
- W3C Standard OWL DL basiert auf der Beschreibungslogik $\mathcal{SHOIN}(\mathcal{D})$
 - wir besprechen zunächst \mathcal{ALC} (Basis für komplexere DLs)

DLs – AUFBAU



- DLs sind eine **Familie** logikbasierter Formalismen zur Wissensrepräsentation
- Spezielle Sprachen v.a. charakterisiert durch:
 - Konstruktoren für komplexe Klassen und Rollen aus einfacheren.
 - Menge von Axiomen um Fakten über Klassen, Rollen und Individuen auszudrücken.
- \mathcal{ALC} ist die kleinste DL, die aussagenlogisch abgeschlossen ist
 - Konjunktion, Disjunktion, Negation sind Konstruktoren, geschrieben \sqcap , \sqcup , \neg .
 - Quantoren schränken Rollenbereiche ein, z.B.:

Man $\sqcap \exists \text{hasChild.Female} \sqcap \exists \text{hasChild.Male} \sqcap \forall \text{hasChild.}(\text{Rich} \sqcup \text{Happy})$

DLs – WEITERE SPRACHMITTEL



- Andere Konstruktoren sind z.B.
 - number restrictions (Kardinalitätseinschränkungen) für Rollen:
 ≥ 3 hasChild, ≤ 1 hasMother
 - qualified number restrictions (klassenspezifische Kardinalitätseinschränkungen) für Rollen:
 ≥ 2 hasChild.Female, ≤ 1 hasParent.Male
 - Nominals (Definition durch Aufzählung):
{Italy, France, Spain}
 - konkrete Bereiche (Datentypen):
hasAge.(≥ 21)
 - inverse Rollen: hasChild⁻
 - transitive Rollen: Trans(hasAncestor)
 - Rollenverknüpfung: Sibling \circ Parent

AGENDA

AIFB 

- Beschreibungslogiken
- **ALC**
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

ALC – GRUNDBAUSTEINE



- Grundbausteine:
 - Klassennamen (auch als Konzepte bezeichnet)
 - Rollennamen
 - Individuennamen

Angabe von Klassen- und Rolleninstanzen:

- Professor(RudiStuder)
 - Individuum RudiStuder ist in Klasse Professor
- Zugehoerigkeit(RudiStuder,AIFB)
 - RudiStuder ist dem AIFB zugehörig

ALC – SUBKLASSENBEZIEHUNGEN

- Professor \sqsubseteq Fakultaetsmitglied
 - „Jeder Professor ist ein Fakultätsmitglied.“
 - entspricht $(\forall x)(\text{Professor}(x) \rightarrow \text{Fakultaetsmitglied}(x))$
 - entspricht owl:subClassOf
- Professor \equiv Fakultaetsmitglied
 - „Die Fakultätsmitglieder sind genau die Professoren.“
 - entspricht $(\forall x)(\text{Professor}(x) \leftrightarrow \text{Fakultaetsmitglied}(x))$
 - entspricht owl:equivalentClass

ALC – KOMPLEXE KLASSEN



- Konjunktion \sqcap entspricht owl:intersectionOf
- Disjunktion \sqcup entspricht owl:unionOf
- Negation \neg entspricht owl:complementOf

Beispiel:

- Professor \sqsubseteq (Person \sqcap Unversitaetsangehoeriger)
 \sqcup (Person \sqcap \neg Doktorand)

$$(\forall x)(\text{Professor}(x) \rightarrow \\ ((\text{Person}(x) \wedge \text{Unversitaetsangehoeriger}(x)) \\ \vee (\text{Person}(x) \wedge \neg \text{Doktorand}(x))))$$

ALC – QUANTOREN AUF ROLLEN

- $\text{Pruefung} \sqsubseteq \forall \text{hatPruefer. Professor}$
 - „Jede Prüfung hat nur Professoren als Prüfer.“
 - $(\forall x)(\text{Pruefung}(x) \rightarrow (\forall y)(\text{hatPruefer}(x,y) \rightarrow \text{Professor}(y)))$
 - entspricht owl:allValuesFrom
- $\text{Pruefung} \sqsubseteq \exists \text{hatPruefer. Person}$
 - „Jede Prüfung hat mindestens einen Prüfer.“
 - $(\forall x)(\text{Pruefung}(x) \rightarrow (\exists y)(\text{hatPruefer}(x,y) \wedge \text{Person}(y)))$
 - entspricht owl:someValuesFrom

WEITERE OWL-KONSTRUKTE IN ALC

AIFB 

- owl:nothing: $\perp \equiv C \sqcap \neg C$
- owl:thing: $\top \equiv C \sqcup \neg C$
- owl:disjointWith:
(gleichbedeutend:) $C \sqcap D \equiv \perp$
 $C \sqsubseteq \neg D$
- rdfs:range: $\top \sqsubseteq \forall R.C$
- rdfs:domain: $\exists R.\top \sqsubseteq C$

ALC – FORMALE SYNTAX



- Folgende Syntaxregeln erzeugen Klassen in \mathcal{ALC} .
Dabei ist A eine atomare Klasse und R eine Rolle.
$$C, D \rightarrow A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$
- Eine \mathcal{ALC} -TBox besteht aus Aussagen der Form $C \sqsubseteq D$ und $C \equiv D$, wobei C, D Klassen sind.
- Eine \mathcal{ALC} -ABox besteht aus Aussagen der Form $C(a)$ und $R(a,b)$, wobei C eine komplexe Klasse, R eine Rolle und a, b Individuen sind.
- Eine \mathcal{ALC} -Wissensbasis besteht aus einer ABox und einer TBox.

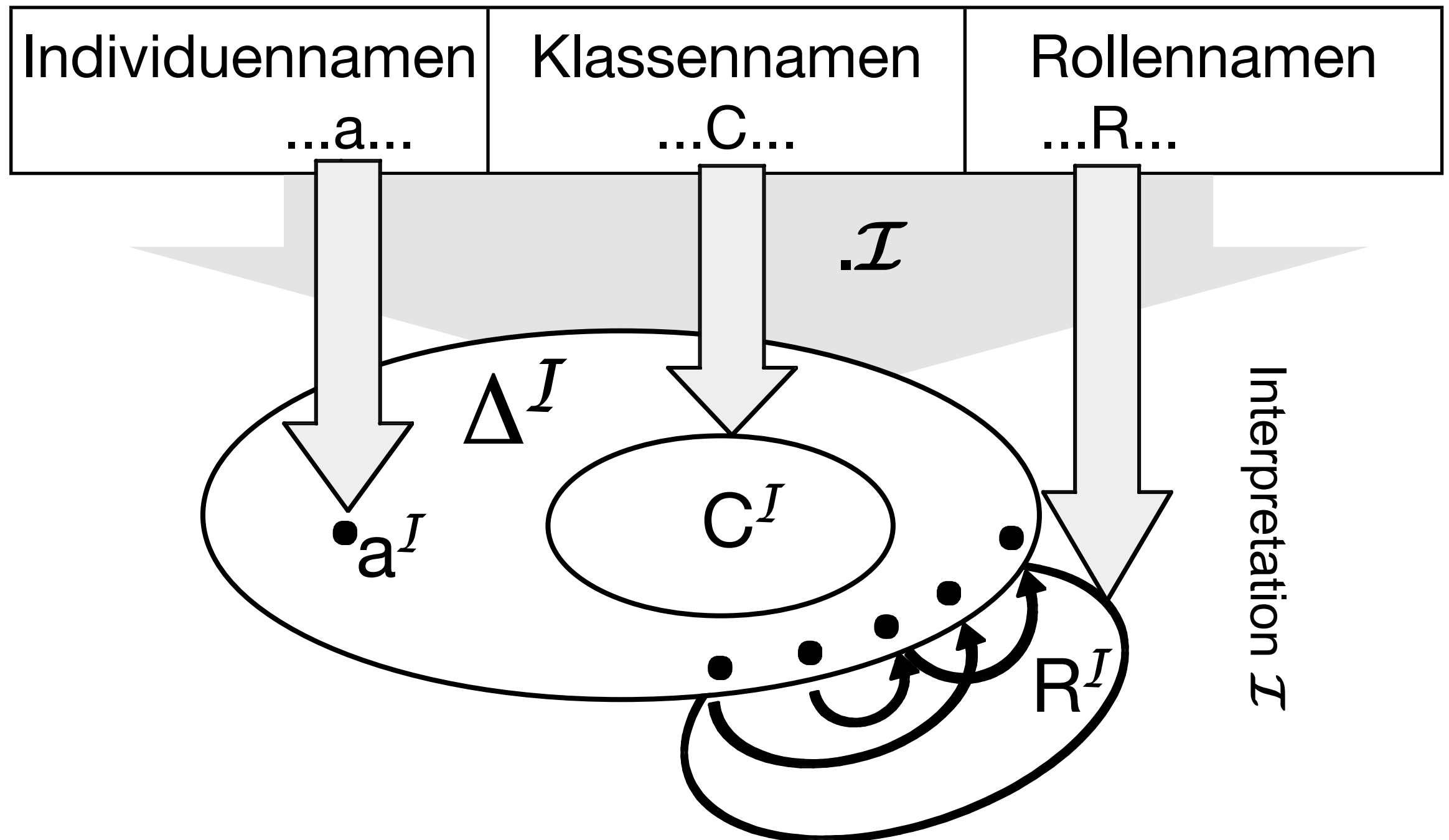
ALC – SEMANTIK (INTERPRETATIONEN)



- wir definieren modelltheoretische Semantik für \mathcal{ALC} (d.h. Folgerung wird über Interpretationen definiert)
- eine Interpretation $\mathcal{I}=(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ besteht aus
 - einer Menge $\Delta^{\mathcal{I}}$, genannt Domäne und
 - einer Funktion $\cdot^{\mathcal{I}}$, die abbildet von
 - ♦ Individuennamen a auf Domänenelemente
 $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
 - ♦ Klassennamen C auf Mengen von Domänenelementen
 $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
 - ♦ Rollennamen R auf Mengen von Paaren von Domänenelementen
 $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

ALC - SEMANTIK (INTERPRETATIONEN)

- schematisch:



ALC-SEMANTIK (KOMPLEXE KLASSEN)

- wird auf komplexe Klassen erweitert:
 - $\top^I = \Delta^I$ $\perp^I = \emptyset$
 - $(C \sqcap D)^I = C^I \cap D^I$ $(C \sqcup D)^I = C^I \cup D^I$
 - $(\neg C)^I = \Delta^I \setminus C^I$
 - $\forall R.C = \{ x \mid \forall (x,y) \in R^I \rightarrow y \in C^I \}$
 - $\exists R.C = \{ x \mid \exists (x,y) \in R^I \text{ mit } y \in C^I \}$
- ...und schließlich auf Axiome:
 - $C(a)$ gilt, wenn $a^I \in C^I$
 - $R(a,b)$ gilt, wenn $(a^I, b^I) \in R^I$
 - $C \sqsubseteq D$ gilt, wenn $C^I \subseteq D^I$
 - $C \equiv D$ gilt, wenn $C^I = D^I$

TBox vs. ABox



Terminologisches Wissen (*TBox*):

$\text{Human} \sqsubseteq \exists \text{hasParent.Human}$

$\text{Orphan} \equiv \text{Human} \sqcap \neg \exists \text{hasParent.Alive}$

Wissen um Individuen (*ABox*):

$\text{Orphan}(\text{harrypotter})$

$\text{hasParent}(\text{harrypotter}, \text{jamespotter})$

Semantik und logische Konsequenzen klar, da gegeben durch formale Semantik.

OWL UND ALC



Folgende OWL DL Sprachelemente sind in ALC repräsentierbar:

- Klassen, Rollen, Individuen
- Klassenzugehörigkeit, Rolleninstanzen
- `owl:Thing` und `owl:Nothing`
- Klasseninklusion, -äquivalenz, -disjunktheit
- `owl:intersectionOf`, `owl:unionOf`
- `owl:complementOf`
- `owl:allValuesFrom`, `owl:someValuesFrom`
- `rdfs:range` und `rdfs:domain`

AGENDA

AIFB 

- Beschreibungslogiken
- ALC
- **OWL als SHOIN(D)**
- Inferenzprobleme
- Tableau-Beweiser

OWL ALS SHOIN(D) - INDIVIDUEN



- owl:sameAs
 - gibt an dass zwei Individuennamen dasselbe Domänenelement bezeichnen
 - DL: $a=b$
 - FOL: Erweiterung durch Gleichheitsprädikat
- owl:differentFrom
 - gibt an dass zwei Individuennamen unterschiedliche Domänenelemente bezeichnen
 - DL: $a \neq b$
 - FOL: $\neg(a=b)$

OWL ALS SHOIN(D) – NOMINALS

Abgeschlossene Klassen

- owl:oneOf
 - definiert eine Klasse durch vollständige Aufzählung ihrer Instanzen
 - DL: $C \equiv \{a, b, c\}$
 - FOL: $(\forall x) (C(x) \leftrightarrow (x=a \vee x=b \vee x=c))$
- owl:hasValue
 - „erzwingt“ Rolle zu einem bestimmten Individuum
 - darstellbar mittels owl:someValuesFrom und owl:oneOf

OWL ALS SHOIN(D) - KARDINALITÄT



Zahlenrestriktionen mittels Gleichheitsprädikat

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxcardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

„Eine Prüfung kann *höchstens* zwei Prüfer haben.“

- DL: $\text{Pruefung} \sqsubseteq \leq 2 \text{ hatPruefer}$
- In FOL: $(P \dots \text{Prüfung}, h \dots \text{hatPruefer})$

$$(\forall x)(P(x) \rightarrow \neg(\exists x_1)(\exists x_2)(\exists x_3)(x_1 \neq x_2 \wedge x_2 \neq x_3 \wedge x_1 \neq x_3 \wedge h(x, x_1) \wedge h(x, x_2) \wedge h(x, x_3)))$$

Entsprechend für die anderen Zahlenrestriktionen

OWL ALS SHOIN(D) – ROLLEN



- `rdfs:subPropertyOf`
 - spezifiziert Unterrolle-Oberrolle-Beziehung
 - DL: $R \sqsubseteq S$
 - FOL: $(\forall x)(\forall y)(R(x,y) \rightarrow S(x,y))$
- Entsprechend Rollenäquivalenz
- inverse Rollen: $R \equiv S^{-}$
 - Konstruktor für Rollen zur Bildung der Inversen
 - FOL: $(\forall x)(\forall y)(R(x,y) \leftrightarrow S(y,x))$
- transitive Rollen: `Trans(R)`
 - gibt an, dass eine Rolle transitiv ist
 - FOL: $(\forall x)(\forall y)(\forall z)(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$
- Symmetrie: $R \equiv R^{-}$
 - ausdrückbar als Rollenäquivalenz mit der Inversen
- Funktionalität: $\top \sqsubseteq \leq 1 R$
 - ausdrückbar durch Klasseninklusion und Kardinalitätsrestriktion
- Inverse Funktionalität: $\top \sqsubseteq \leq 1 R^{-}$
 - ausdrückbar wie Funktionalität zuzüglich inverser Rolle

OWL ALS SHOIN(D) – ÜBERBLICK



Erlaubt sind:

- ALC
- Gleichheit und Ungleichheit zwischen Individuen
- Abgeschlossene Klassen
- Zahlenrestriktionen
- Subrollen und Rollenäquivalenz
- Inverse und transitive Rollen
- Datentypen

DL-SYNTAX - ÜBERSICHT



Concepts		
\mathcal{ALC}	Atomic	A, B
	Not	$\neg C$
	And	$C \sqcap D$
	Or	$C \sqcup D$
	Exists	$\exists R.C$
	For all	$\forall R.C$
$\mathcal{Q}(\mathcal{N})$	At least	$\geq n \ R.C \ (\geq n \ R)$
	At most	$\leq n \ R.C \ (\leq n \ R)$
\mathcal{O}	Nominal	$\{i_1, \dots, i_n\}$

Roles		
\mathcal{I}	Atomic	R
	Inverse	R^-

Ontology (=Knowledge Base)

Concept Axioms (TBox)

Subclass	$C \sqsubseteq D$
Equivalent	$C \equiv D$

Role Axioms (RBox)

\mathcal{H}	Subrole	$R \sqsubseteq S$
\mathcal{S}	Transitivity	$\text{Trans}(S)$

Assertional Axioms (ABox)

Instance	$C(a)$
Role	$R(a, b)$
Same	$a = b$
Different	$a \neq b$

$S = \mathcal{ALC} + \text{Transitivity}$ **OWL DL = $\mathcal{SHOIN}(\mathcal{D})$** (D: concrete domain)

DL-SYNTAX - KLASSENKONSTRUKTOREN

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male	$C_1(x) \wedge \dots \wedge C_n(x)$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer	$C_1(x) \vee \dots \vee C_n(x)$
complementOf	$\neg C$	\neg Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}	$x = x_1 \vee \dots \vee x = x_n$
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor	$\forall y.P(x, y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer	$\exists y.P(x, y) \wedge C(y)$
maxCardinality	$\leq nP$	≤ 1 hasChild	$\exists \leq n y.P(x, y)$
minCardinality	$\geq nP$	≥ 2 hasChild	$\exists \geq n y.P(x, y)$

Beliebig komplexes Schachteln von Konstruktoren erlaubt:

Person $\sqcap \forall$ hasChild.(Doctor $\sqcup \exists$ hasChild.Doctor)

DL-SYNTAX - AXIOME



Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
sameAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} \equiv {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg\{peter\}$
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	cost \equiv price
inverseOf	$P_1 \equiv P_2^-$	hasChild \equiv hasParent ⁻
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor
functionalProperty	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1$ hasSSN ⁻

- General Class Inclusion (\sqsubseteq) genügt:

$$C \equiv D \text{ gdw } (C \sqsubseteq D \text{ und } D \sqsubseteq C)$$
- Offensichtliche FOL-Äquivalenzen

$$C \equiv D \Leftrightarrow (\forall x) (C(x) \leftrightarrow D(x))$$

$$C \sqsubseteq D \Leftrightarrow (\forall x) (C(x) \rightarrow D(x))$$

WISSENSMODELLIERUNG OWA vs. CWA

AIFB 

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

WISSENSMODELLIERUNG OWA vs. CWA

AIFB 

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

child(Bill,Bob)
Man(Bob)

WISSENSMODELLIERUNG OWA vs. CWA

AIFB 

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

*Are all children of Bill
male?*

child(Bill,Bob)

Man(Bob)

? $\models \forall \text{child.Man(Bill)}$

WISSENSMODELLIERUNG OWA vs. CWA

AIFB 

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

No idea, since we do not know all children of Bill.

Are all children of Bill male?

child(Bill,Bob)

Man(Bob)

? $\models \forall \text{child.Man(Bill)}$

DL answers

don't know

WISSENSMODELLIERUNG OWA vs. CWA



OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

	Are all children of Bill male?	No idea, since we do not know all children of Bill.	If we assume that we know everything about Bill, then all of his children are male.
child(Bill,Bob)		DL answers	Prolog
Man(Bob)	? $\models \forall \text{child.Man(Bill)}$	don't know	yes

WISSENSMODELLIERUNG OWA vs. CWA

AIFB 

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

Are all children of Bill male?

No idea, since we do not know all children of Bill.

If we assume that we know everything about Bill, then all of his children are male.

child(Bill,Bob)

Man(Bob)

≤ 1 child.T(Bill)

? $\models \forall \text{child.Man(Bill)}$

? $\models \forall \text{child.Man(Bill)}$

DL answers

don't know

Prolog

yes

WISSENSMODELLIERUNG OWA vs. CWA

AIFB 

OWA: Open World Assumption

Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird.

OWL verwendet OWA!

CWA: Closed World Assumption

Es wird angenommen, dass die Wissensbasis alle Individuen enthält.

Are all children of Bill male?

No idea, since we do not know all children of Bill.

If we assume that we know everything about Bill, then all of his children are male.

child(Bill,Bob)

Man(Bob)

≤ 1 child.T(Bill)

? $\models \forall \text{child.Man(Bill)}$

? $\models \forall \text{child.Man(Bill)}$

DL answers

don't know

yes

Prolog

yes

Now we know everything about Bill's children.

AGENDA

AIFB 

- Beschreibungslogiken
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- Tableau-Beweiser

WICHTIGE INFERENZPROBLEME

WICHTIGE INFERENZPROBLEME



- Globale Konsistenz der Wissensbasis
 - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz
 - Muss Klasse C leer sein?

$KB \models \text{false}?$

$C \equiv \perp?$

WICHTIGE INFERENZPROBLEME



- Globale Konsistenz der Wissensbasis
 - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz
 - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption)
 - Strukturierung der Wissensbasis
- Klassenäquivalenz
 - Sind zwei Klassen eigentlich dieselbe?

$KB \models \text{false?}$

$C \equiv \perp?$

$C \sqsubseteq D?$

$C \equiv D?$

WICHTIGE INFERENZPROBLEME



- Globale Konsistenz der Wissensbasis
 - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz
 - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption)
 - Strukturierung der Wissensbasis
- Klassenäquivalenz
 - Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit
 - Sind zwei Klassen disjunkt?
- Klassenzugehörigkeit
 - Ist Individuum a in der Klasse C?

$KB \models \text{false?}$

$C \equiv \perp?$

$C \sqsubseteq D?$

$C \equiv D?$

$C \sqcap D = \perp?$

$C(a)?$

WICHTIGE INFERENZPROBLEME



- Globale Konsistenz der Wissensbasis $KB \models \text{false?}$
 - Ist Wissensbasis sinnvoll?
- Klassenkonsistenz $C \equiv \perp?$
 - Muss Klasse C leer sein?
- Klasseninklusion (Subsumption) $C \sqsubseteq D?$
 - Strukturierung der Wissensbasis
- Klassenäquivalenz $C \equiv D?$
 - Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit $C \sqcap D = \perp?$
 - Sind zwei Klassen disjunkt?
- Klassenzugehörigkeit $C(a)?$
 - Ist Individuum a in der Klasse C?
- Instanzgenerierung (Retrieval) „alle X mit C(X) finden“
 - Finde alle (bekannten!) Individuen zur Klasse C.

ENTSCHEIDBARKEIT VON OWL DL

ENTSCHEIDBARKEIT VON OWL DL



- Entscheidbarkeit: zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.

ENTSCHEIDBARKEIT VON OWL DL



- Entscheidbarkeit: zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.
- OWL DL ist Fragment von FOL, also könnten (im Prinzip) FOL-Inferenzalgorithmen (Resolution, Tableaux) verwendet werden.

ENTSCHEIDBARKEIT VON OWL DL



- Entscheidbarkeit: zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.
- OWL DL ist Fragment von FOL, also könnten (im Prinzip) FOL-Inferenzalgorithmen (Resolution, Tableaux) verwendet werden.
- Diese terminieren aber nicht immer!

ENTSCHEIDBARKEIT VON OWL DL



- Entscheidbarkeit: zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.
- OWL DL ist Fragment von FOL, also könnten (im Prinzip) FOL-Inferenzalgorithmen (Resolution, Tableaux) verwendet werden.
- Diese terminieren aber nicht immer!

ENTSCHEIDBARKEIT VON OWL DL



- Entscheidbarkeit: zu jedem Inferenzproblem gibt es einen immer terminierenden Algorithmus.
- OWL DL ist Fragment von FOL, also könnten (im Prinzip) FOL-Inferenzalgorithmen (Resolution, Tableaux) verwendet werden.
- Diese terminieren aber nicht immer!
- Problem: Finde immer terminierende Algorithmen!
Keine „naiven“ Lösungen in Sicht!

RÜCKFÜHRUNG AUF UNERFÜLLBARKEIT



- Wir werden das Tableauverfahren für OWL DL abwandeln.
 - Genauer: Wir werden nur \mathcal{ALC} behandeln.
 - Tableauverfahren zeigt Unerfüllbarkeit einer Theorie.
- Rückführung der Inferenzprobleme auf das Finden von Inkonsistenten in der Wissensbasis, d.h. zeigen der Unerfüllbarkeit der Wissensbasis!

RÜCKFÜHRUNG AUF UNERFÜLLBARKEIT



- Klassenkonsistenz $C \equiv \perp$ gdw
 - $KB \cup \{C(a)\}$ unerfüllbar (a neu)
- Klasseninklusion (Subsumption) $C \sqsubseteq D$ gdw
 - $KB \cup \{C \sqcap \neg D(a)\}$ unerfüllbar (a neu)
- Klassenäquivalenz $C \equiv D$ gdw
 - $C \sqsubseteq D$ und $D \sqsubseteq C$
- Klassendisjunktheit $C \sqcap D = \perp$ gdw
 - $KB \cup \{(C \sqcap D)(a)\}$ unerfüllbar (a neu)
- Klassenzugehörigkeit $C(a)$ gdw
 - $KB \cup \{\neg C(a)\}$ unerfüllbar (a neu)
- Instanzgenerierung (Retrieval) alle $C(X)$ finden
 - Prüfe Klassenzugehörigkeit für alle Individuen.
 - Schwerer, dies gut zu implementieren!

AGENDA

AIFB 

- Beschreibungslogiken
- ALC
- OWL als SHOIN(D)
- Inferenzprobleme
- **Tableau-Beweiser**

TABLEAU - TRANSFORMATION IN NNF

Gegeben eine Wissensbasis W .

- Ersetze $C \equiv D$ durch $C \sqsubseteq D$ und $D \sqsubseteq C$.
- Ersetze $C \sqsubseteq D$ durch $\neg C \sqcup D$.
- Wende die Regeln auf der folgenden Folie an, bis es nicht mehr geht.

Resultierende Wissensbasis: $NNF(W)$

Negationsnormalform von W .

Negation steht nur noch direkt vor atomaren Klassen.

TABLEAU - TRANSFORMATION IN NNF

AIFB 

$NNF(C) = C$, falls C atomar ist

$NNF(\neg C) = \neg C$, falls C atomar ist

$NNF(\neg\neg C) = NNF(C)$

$NNF(C \sqcup D) = NNF(C) \sqcup NNF(D)$

$NNF(C \sqcap D) = NNF(C) \sqcap NNF(D)$

$NNF(\neg(C \sqcup D)) = NNF(\neg C) \sqcap NNF(\neg D)$

$NNF(\neg(C \sqcap D)) = NNF(\neg C) \sqcup NNF(\neg D)$

$NNF(\forall R.C) = \forall R.NNF(C)$

$NNF(\exists R.C) = \exists R.NNF(C)$

$NNF(\neg\forall R.C) = \exists R.NNF(\neg C)$

$NNF(\neg\exists R.C) = \forall R.NNF(\neg C)$

W und $NNF(W)$ sind logisch äquivalent.

TABLEAU - NNF - BEISPIEL

AIFB 

$$P \sqsubseteq (E \sqcap U) \sqcup \neg(\neg E \sqcup D).$$

In Negationsnormalform:

$$\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D).$$

NAIVES TABLEAUFERFAHREN



Rückführung auf Unerfüllbarkeit/Widerspruch

Idee:

- Gegeben Wissensbasis W .
- Erzeugen von Konsequenzen der Form $C(a)$ und $\neg C(a)$, bis Widerspruch gefunden.

TABLEAU – EINFACHES BEISPIEL

AIFB 

$C(a)$

$(\neg C \sqcap D)(a)$

$\neg C(a)$ ist logische Konsequenz:

2. Formel in FOL: $\neg C(a) \wedge D(a)$

daraus folgt u.a. $\neg C(a)$

Widerspruch ist gefunden.

TABLEAU - WEITERES BEISPIEL

Teilen des Tableaus in zwei *Zweige*.

TABLEAU – WEITERES BEISPIEL

AIFB  $C(a)$ $\neg C \sqcup D$ $\neg D(a)$

Ableitung von Konsequenzen:

 $C(a)$

Teilen des Tableaus in zwei *Zweige*.

TABLEAU – WEITERES BEISPIEL

AIFB  $C(a)$ $\neg C \sqcup D$ $\neg D(a)$

Ableitung von Konsequenzen:

 $C(a)$ $\neg D(a)$

Teilen des Tableaus in zwei *Zweige*.

TABLEAU – WEITERES BEISPIEL

AIFB  $C(a)$ $\neg C \sqcup D$ $\neg D(a)$

Ableitung von Konsequenzen:

 $C(a)$ $\neg D(a)$ $(\neg C \sqcup D)(a)$

Nun Fallunterscheidung

Teilen des Tableaus in zwei *Zweige*.

TABLEAU - WEITERES BEISPIEL

AIFB 

$C(a)$

$\neg C \sqcup D$

$\neg D(a)$

Ableitung von Konsequenzen:

$C(a)$

$\neg D(a)$

$(\neg C \sqcup D)(a)$

Nun Fallunterscheidung

1. $\neg C(a)$

Widerspruch

Teilen des Tableaus in zwei *Zweige*.

TABLEAU - WEITERES BEISPIEL



$C(a)$

$\neg C \sqcup D$

$\neg D(a)$

Ableitung von Konsequenzen:

$C(a)$

$\neg D(a)$

$(\neg C \sqcup D)(a)$

Nun Fallunterscheidung

1. $\neg C(a)$

Widerspruch

2. $D(a)$

Widerspruch

Teilen des Tableaus in zwei *Zweige*.

TABLEAU – DEFINITIONEN



- *Tableauzweig*:
Endliche Menge von Aussagen der Form
 $C(a)$, $\neg C(a)$, $R(a,b)$.
- *Tableau*: Endliche Menge von Tableauzweigen.
- Tableauzweig ist *abgeschlossen* wenn er ein Paar widersprüchlicher Aussagen $C(a)$ und $\neg C(a)$ enthält.
- Tableau ist *abgeschlossen*, wenn jeder Zweig von ihm abgeschlossen ist.

TABLEAU - ERZEUGUNG

AIFB 

Auswahl	Aktion
$C(a) \in W$ (ABox)	Füge $C(a)$ hinzu.
$R(a, b) \in W$ (ABox)	Füge $R(a, b)$ hinzu.
$C \in W$ (TBox)	Füge $C(a)$ für ein bekanntes Individuum a hinzu.
$(C \sqcap D)(a) \in A$	Füge $C(a)$ und $D(a)$ hinzu.
$(C \sqcup D)(a) \in A$	Dupliziere den Zweig. Füge zum einen Zweig $C(a)$ und zum anderen Zweig $D(a)$ hinzu.
$(\exists R.C)(a) \in A$	Füge $R(a, b)$ und $C(b)$ für neues Individuum b hinzu.
$(\forall R.C)(a) \in A$	Falls $R(a, b) \in A$, so füge $C(b)$ hinzu.

TABLEAU - ERZEUGUNG

AIFB 

Auswahl	Aktion
$C(a) \in W$ (ABox)	Füge $C(a)$ hinzu.
$R(a, b) \in W$ (ABox)	Füge $R(a, b)$ hinzu.
$C \in W$ (TBox)	Füge $C(a)$ für ein bekanntes Individuum a hinzu.
$(C \sqcap D)(a) \in A$	Füge $C(a)$ und $D(a)$ hinzu.
$(C \sqcup D)(a) \in A$	Dupliziere den Zweig. Füge zu einem Zweig $C(a)$ und zum anderen Zweig $D(a)$ hinzu.
$(\exists R.C)(a) \in A$	Füge $R(a, b)$ und $C(b)$ für neues Individuum b hinzu.
$(\forall R.C)(a) \in A$	Falls $R(a, b) \in A$, so füge $C(b)$ hinzu.

Ist das resultierende Tableau abgeschlossen, so ist die ursprüngliche Wissensbasis unerfüllbar.

Man wählt dabei immer nur solche Elemente aus, die auch wirklich zu neuen Elementen im Tableau führen. Ist dies nicht möglich, so terminiert der Algorithmus und die Wissensbasis ist erfüllbar.

TABLEAU - BEISPIEL (1/2)

AIFB 

- P ... Professor
E ... Person
U ... Universitätsangehöriger
D ... Doktorand
- Wissensbasis: $P \sqsubseteq (E \sqcap U) \sqcup (E \sqcap \neg D)$
Ist $P \sqsubseteq E$ logische Konsequenz?
- Wissensbasis (mit Anfrage) in NNF:
 $\{\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D), (P \sqcap \neg E)(a)\}$

TABLEAU - BEISPIEL (2/2)

AIFB 

TBox: $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$

Tableau:

$(P \sqcap \neg E)(a)$ (aus Wissensbasis)

$P(a)$

$\neg E(a)$

$(\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

TABLEAU - BEISPIEL (2/2)

AIFB 

TBox: $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$

Tableau:

$(P \sqcap \neg E)(a)$ (aus Wissensbasis)

$P(a)$

$\neg E(a)$

$(\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$\neg P(a)$ $((E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

TABLEAU - BEISPIEL (2/2)

AIFB 

TBox: $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$

Tableau:

$(P \sqcap \neg E)(a)$ (aus Wissensbasis)

$P(a)$

$\neg E(a)$

$(\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$\neg P(a)$ $((E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$(E \sqcap U)(a)$

$(E \sqcap \neg D)(a)$

$E(a)$

$E(a)$

$U(a)$

$\neg D(a)$

TABLEAU - BEISPIEL (2/2)

AIFB 

TBox: $\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D)$

Tableau:

$(P \sqcap \neg E)(a)$ (aus Wissensbasis)

$P(a)$

$\neg E(a)$

$(\neg P \sqcup (E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$\neg P(a)$ $((E \sqcap U) \sqcup (E \sqcap \neg D))(a)$

$(E \sqcap U)(a)$

$(E \sqcap \neg D)(a)$

$E(a)$

$E(a)$

$U(a)$

$\neg D(a)$

D.h. Wissensbasis ist unerfüllbar, d.h. $P \sqsubseteq E$.

TABLEAU - TERMINIERUNGSPROBLEM



- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

TABLEAU - TERMINIERUNGSPROBLEM



- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

Person(Bill)

$\neg \text{Person} \sqcup \exists \text{hasParent. Person}(\text{Bill})$

TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

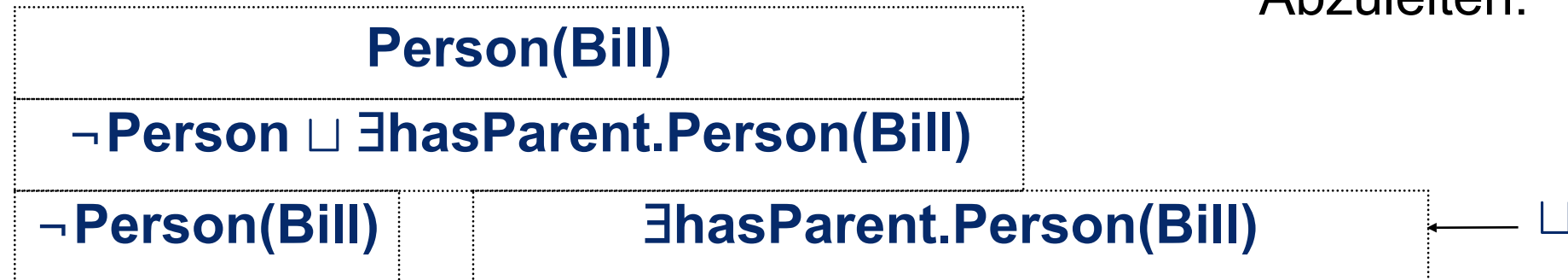


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

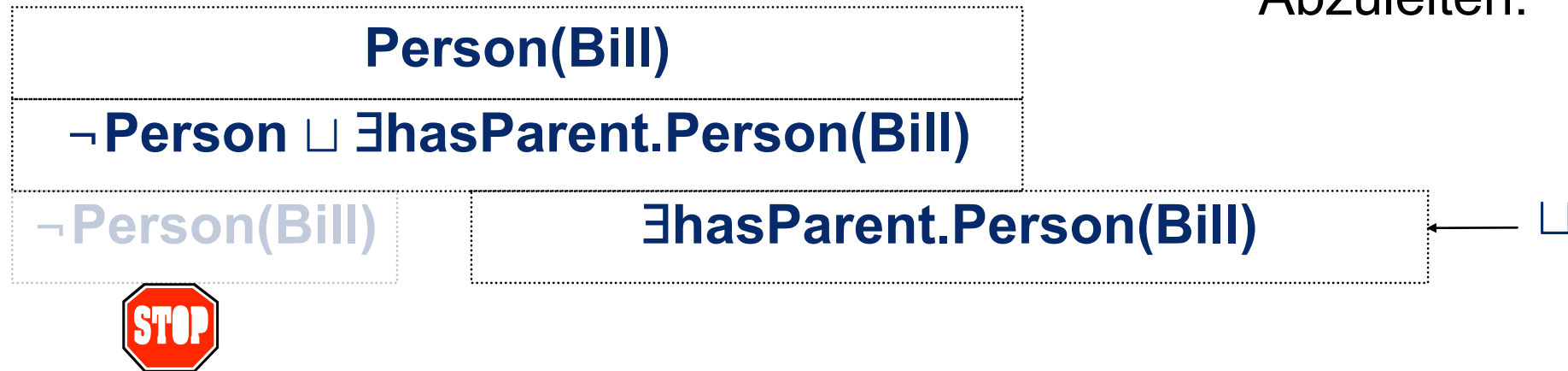


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

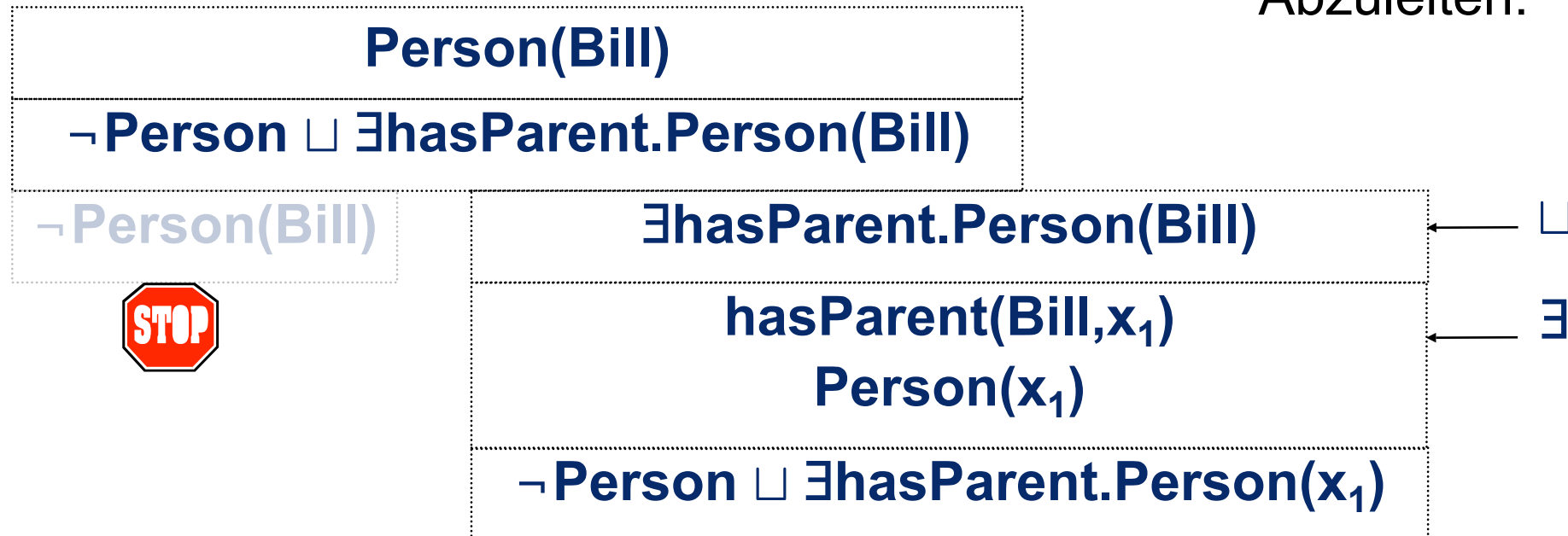


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

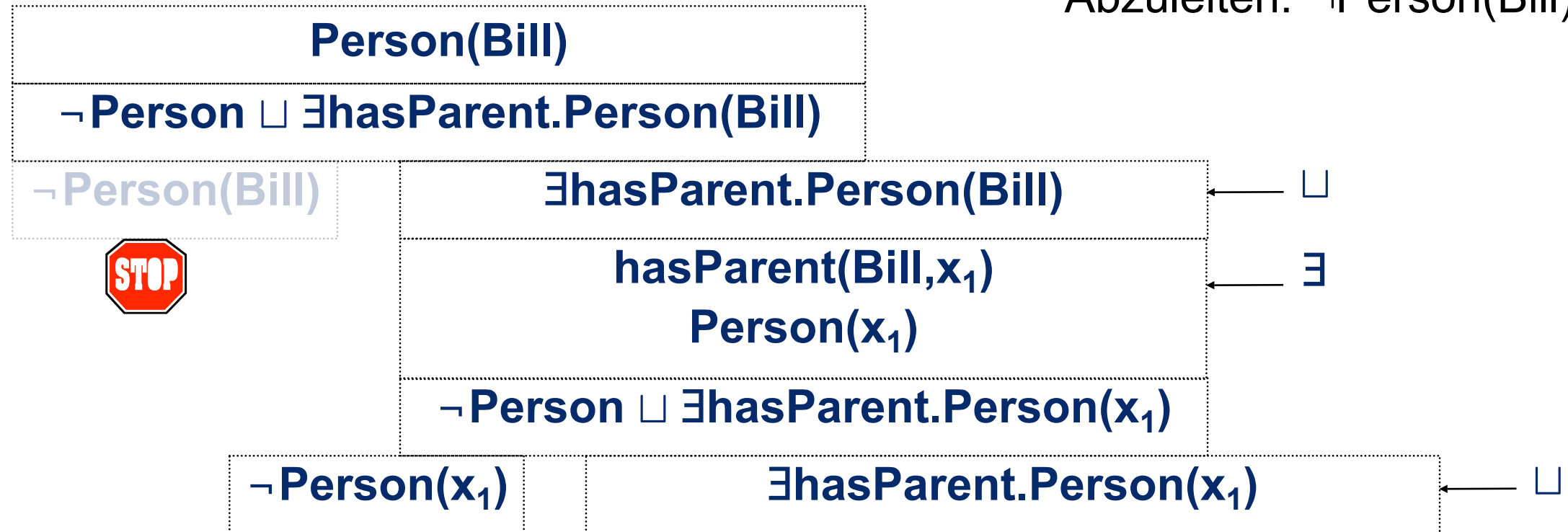


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

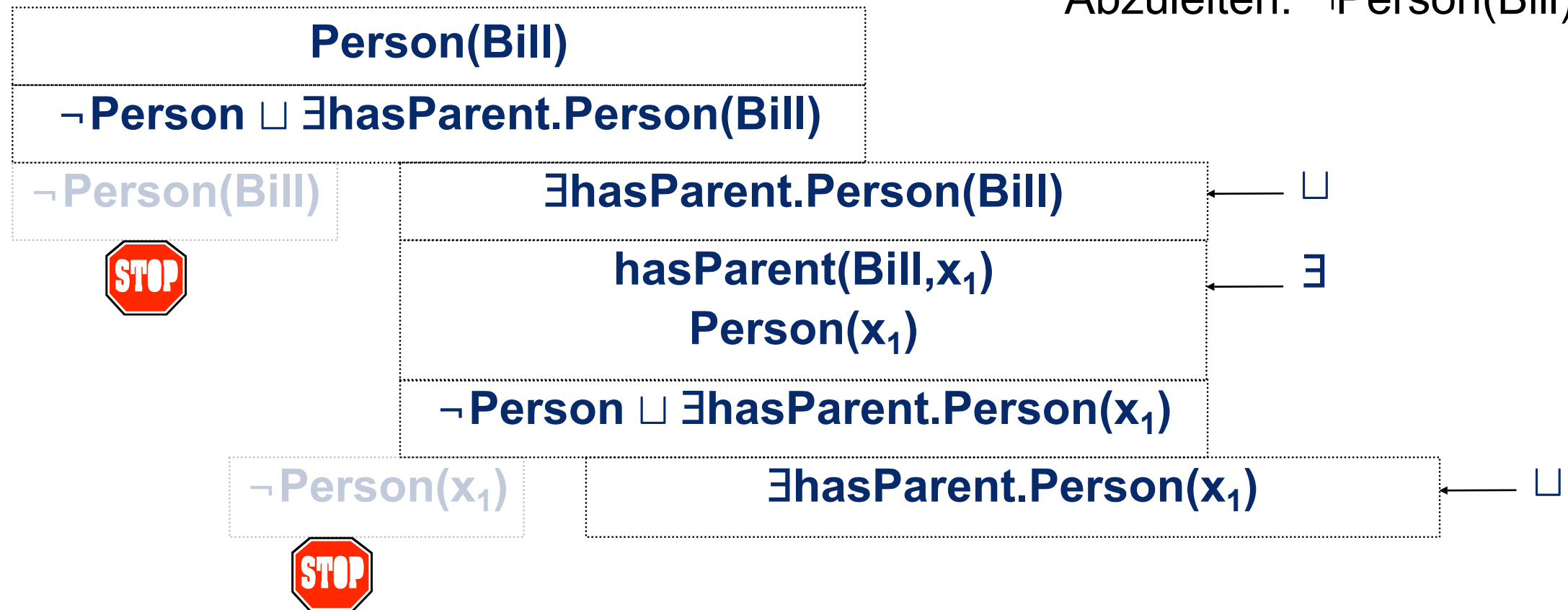


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

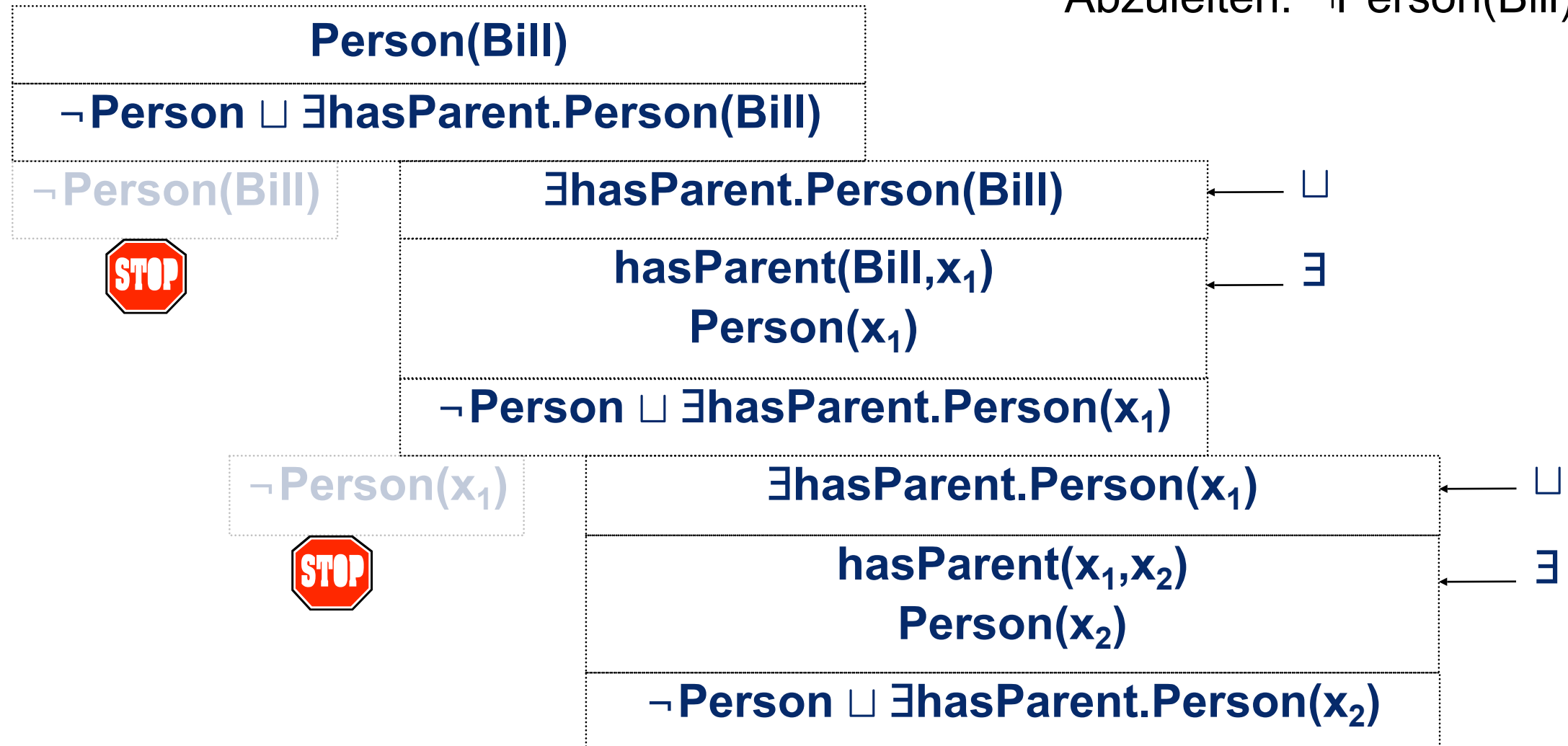


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

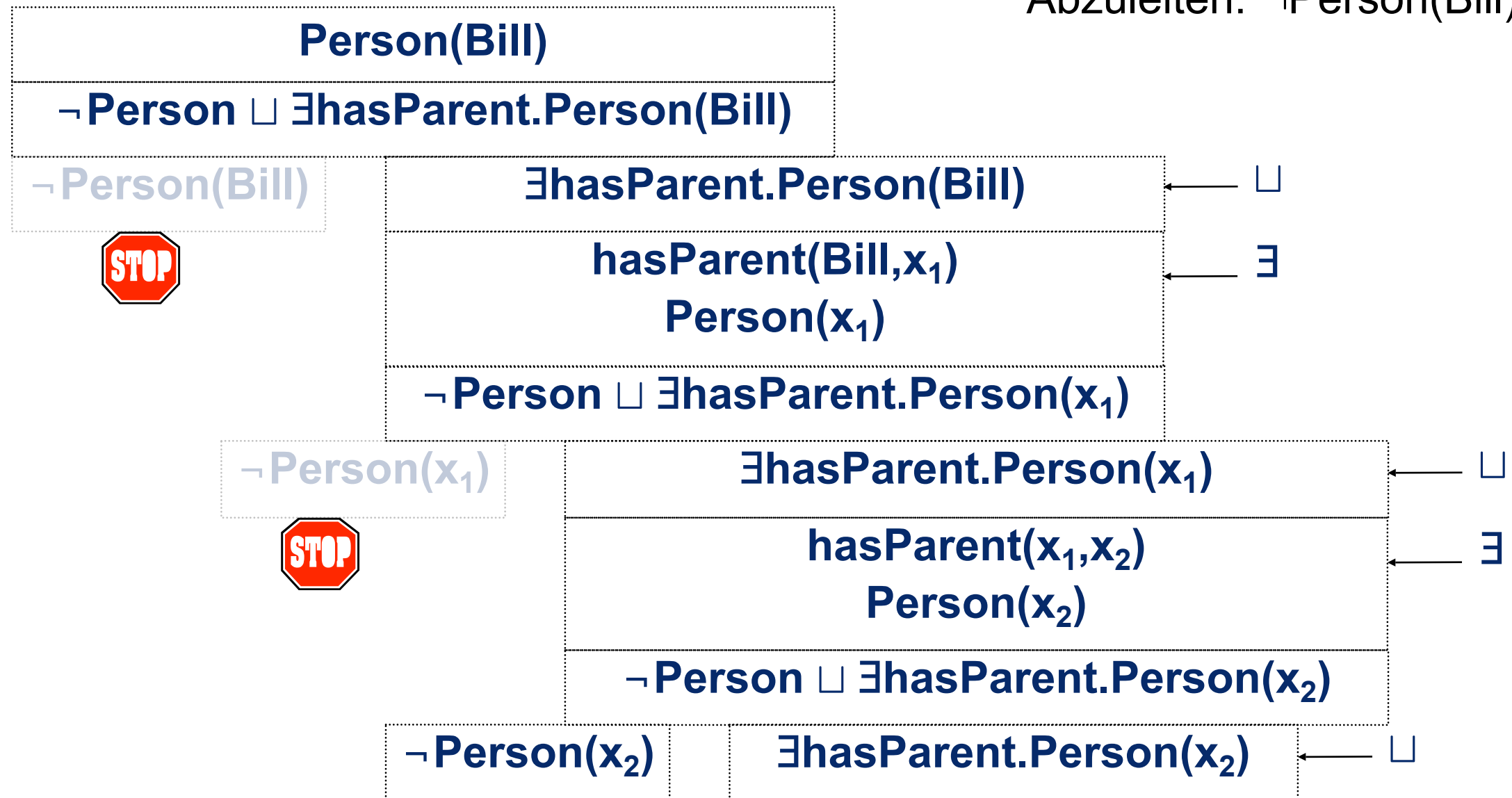


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

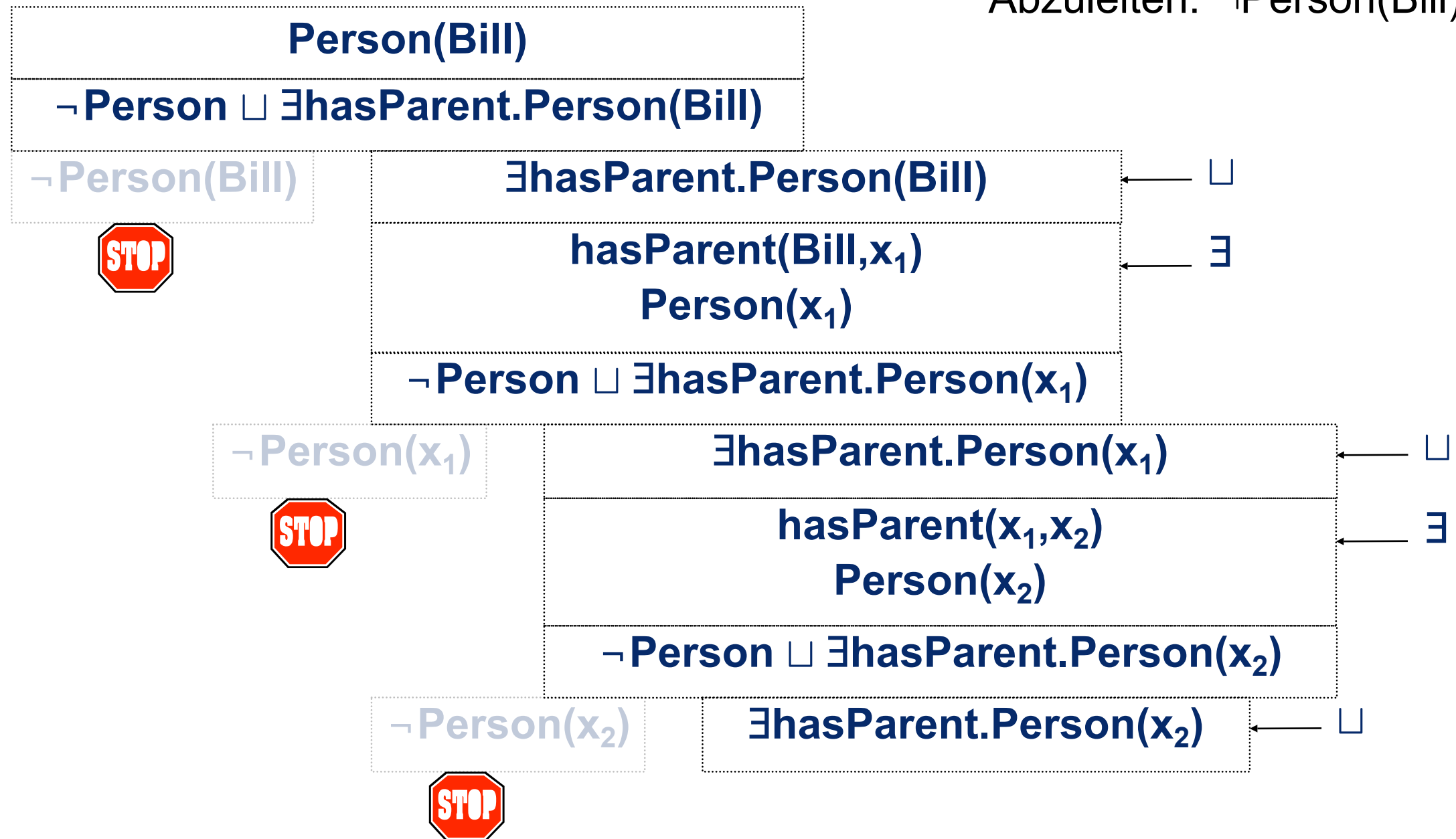


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

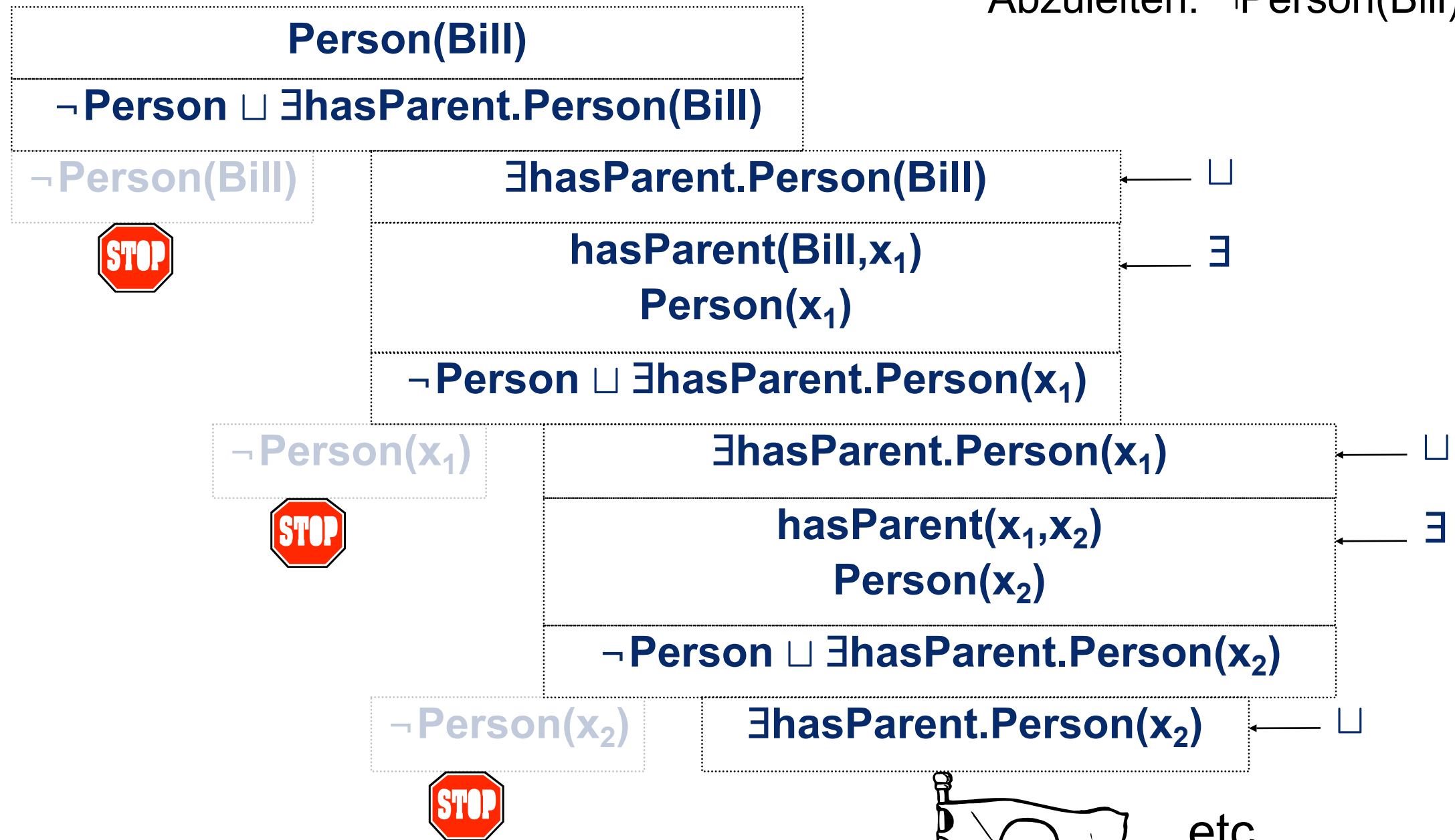
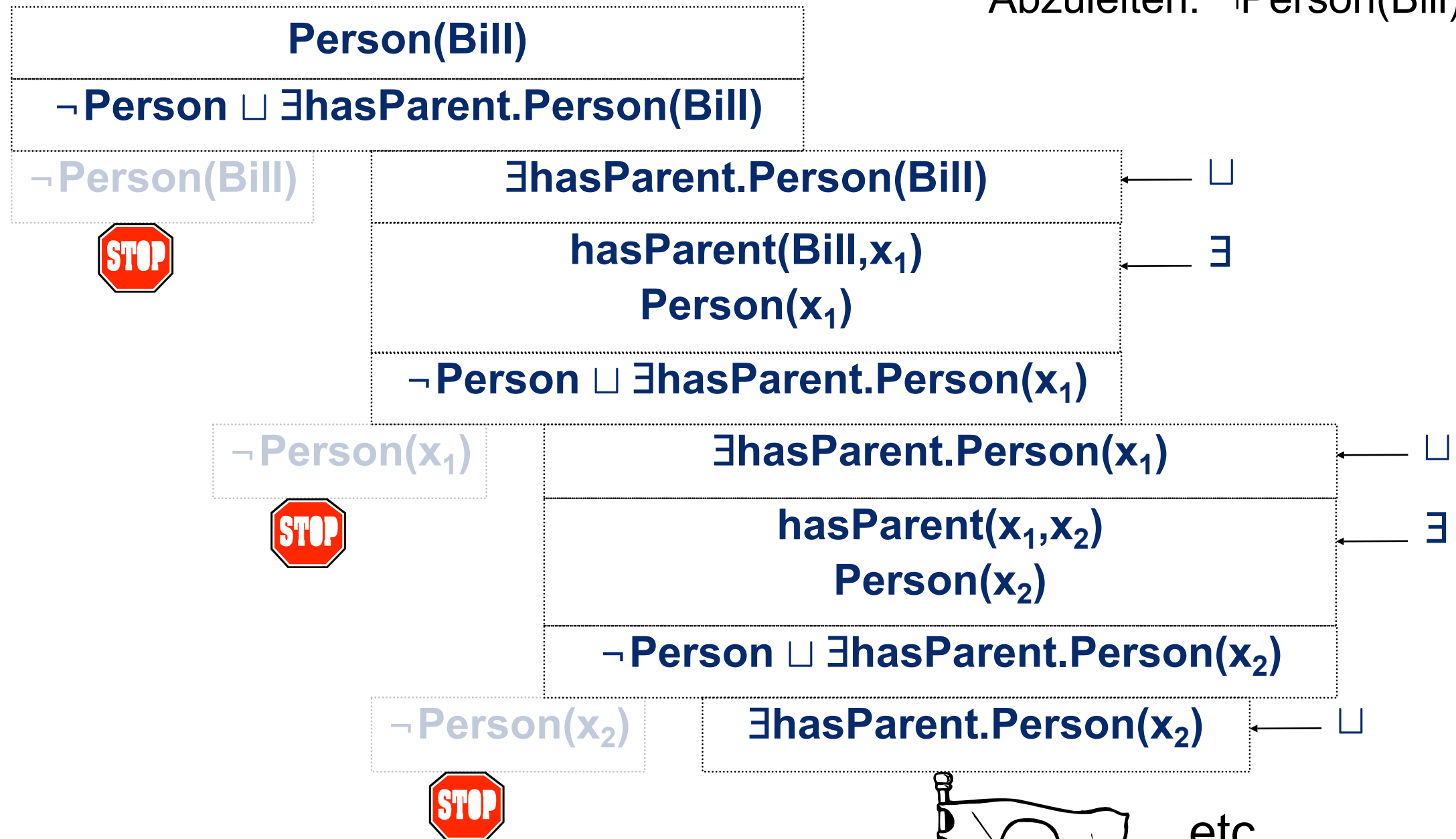


TABLEAU - TERMINIERUNGSPROBLEM

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$



**Problem tritt auf durch
Existenzquantoren (und minCardinality)**

TABLEAU - BLOCKING - IDEE

- AIFB  • Wir haben folgendes konstruiert:

TABLEAU - BLOCKING - IDEE

- AIFB  • Wir haben folgendes konstruiert:

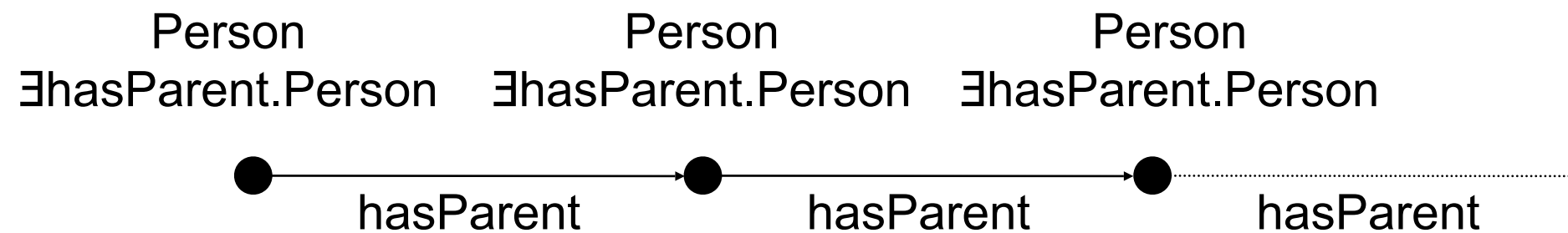
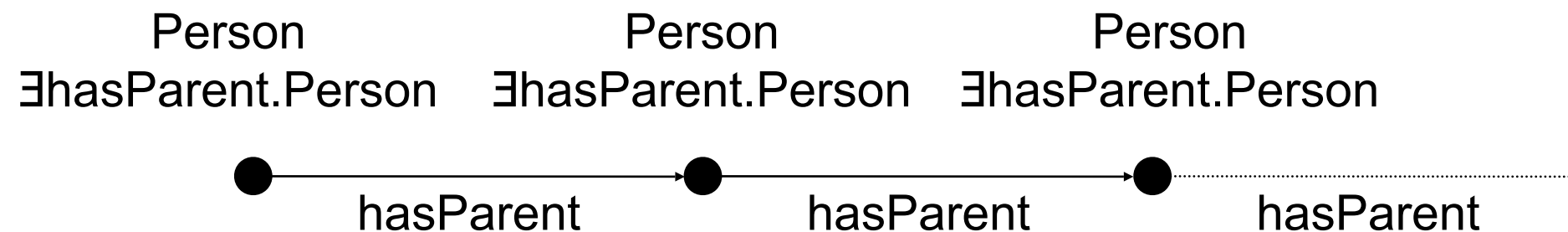


TABLEAU - BLOCKING - IDEE

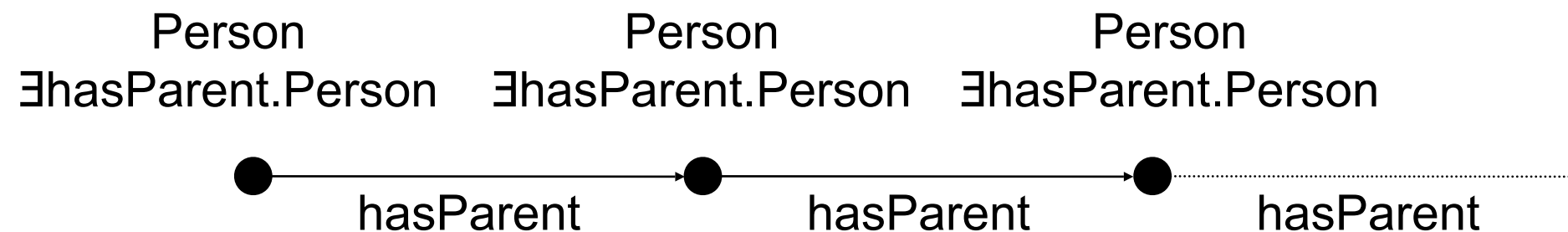
- AIFB  • Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:

TABLEAU - BLOCKING - IDEE

- AIFB  • Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:

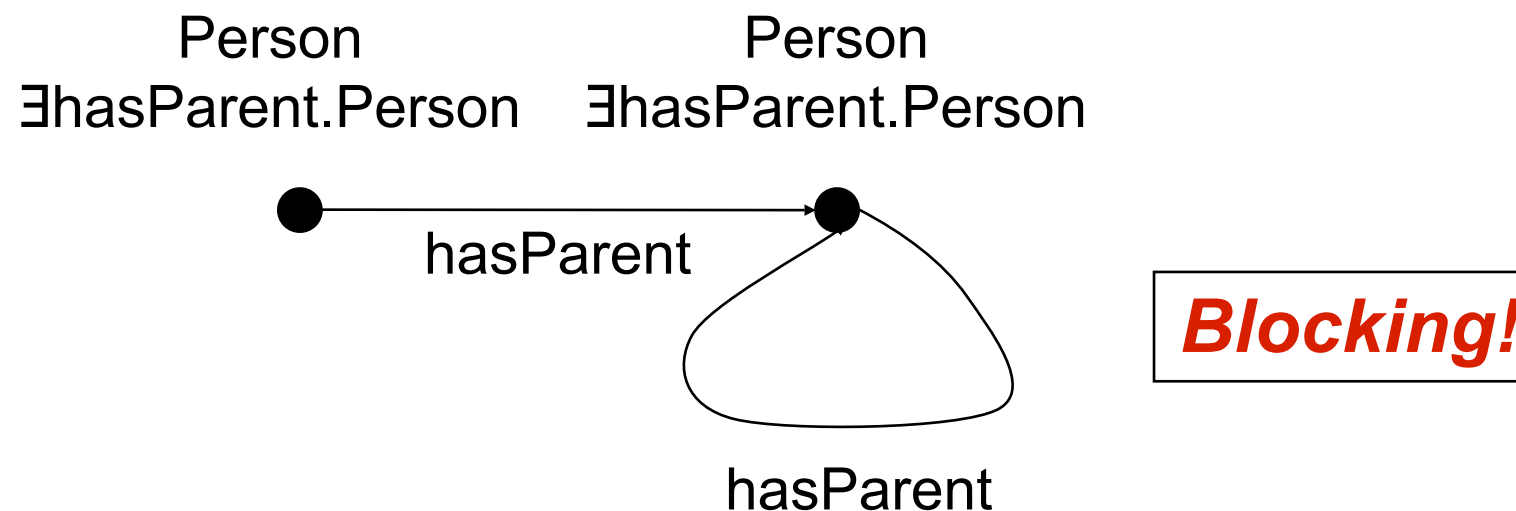
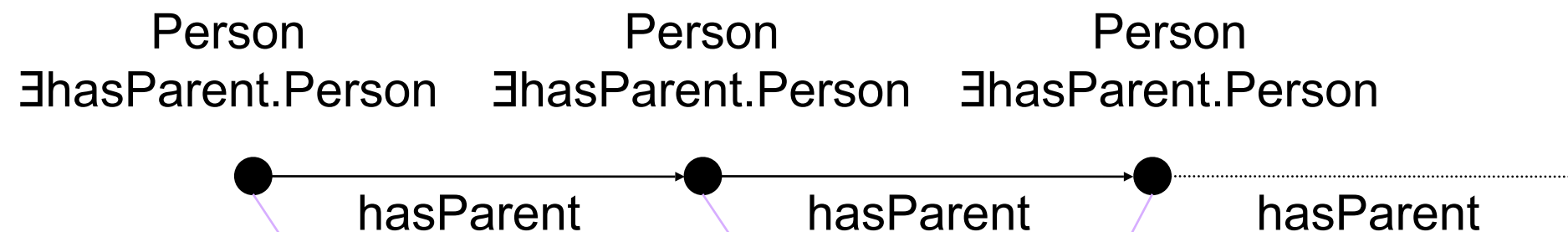
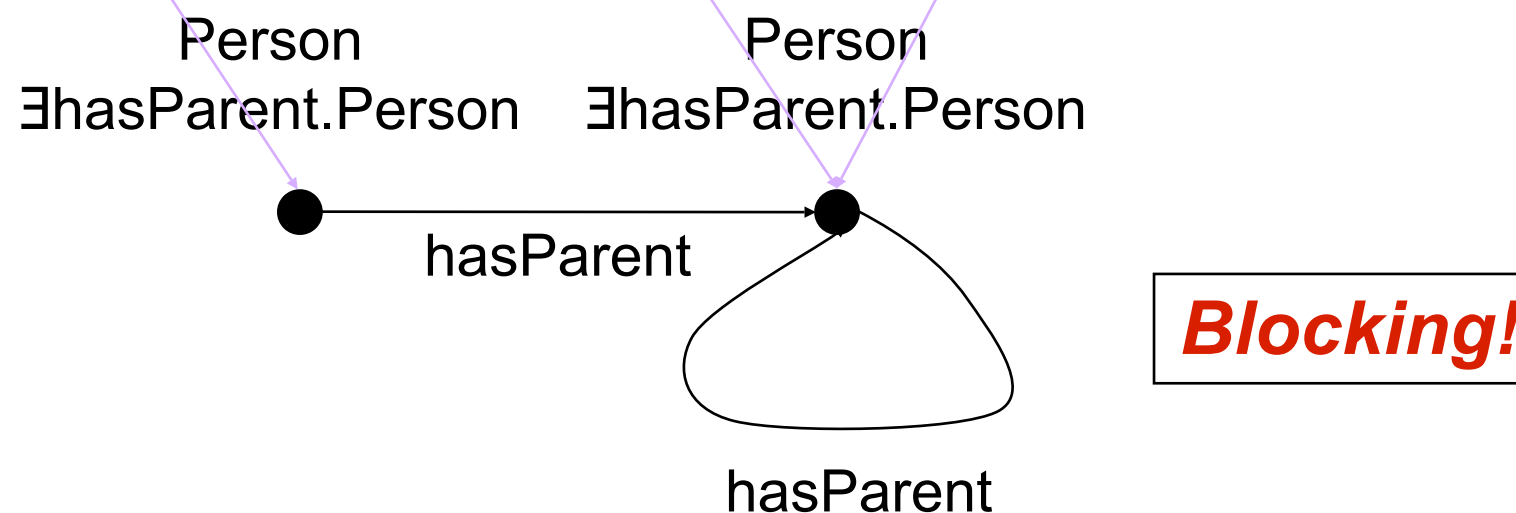


TABLEAU - BLOCKING - IDEE

- AIFB  • Wir haben folgendes konstruiert:



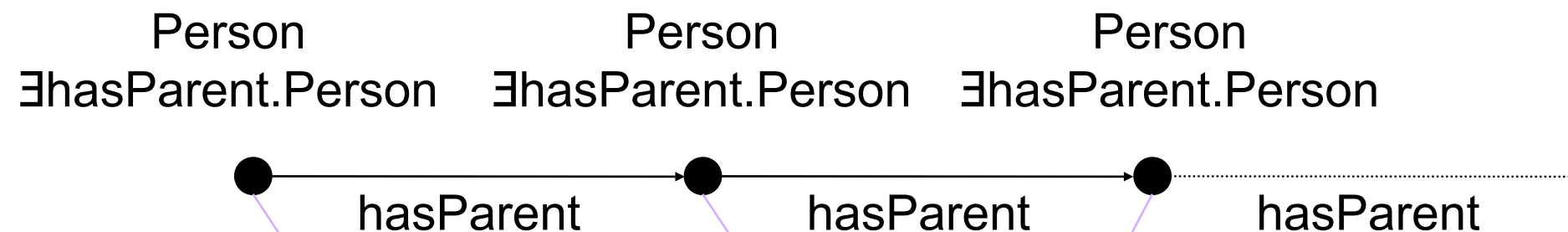
- Folgendes wäre aber auch denkbar:



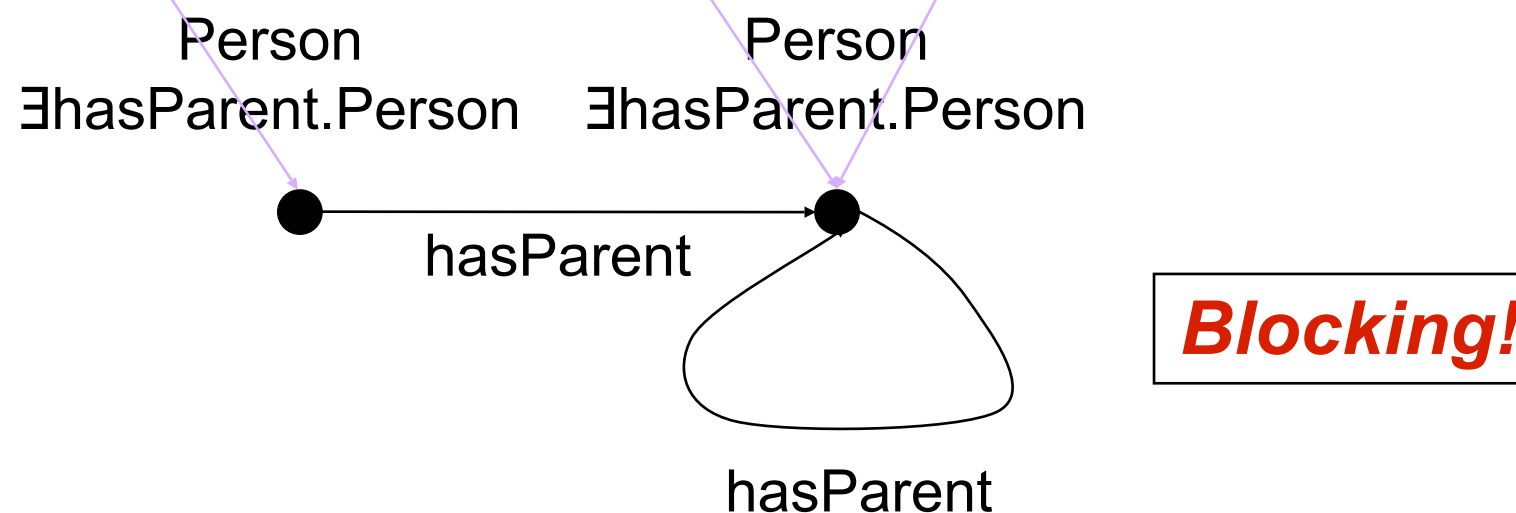
Blocking!

TABLEAU - BLOCKING - IDEE

- AIFB  • Wir haben folgendes konstruiert:



- Folgendes wäre aber auch denkbar:



D.h. Wiederverwendung alter Knoten!

Es muss natürlich formal nachgewiesen werden, dass das ausreicht!

TABLEAU MIT BLOCKING



- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent. Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

Person(Bill)

TABLEAU MIT BLOCKING

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$

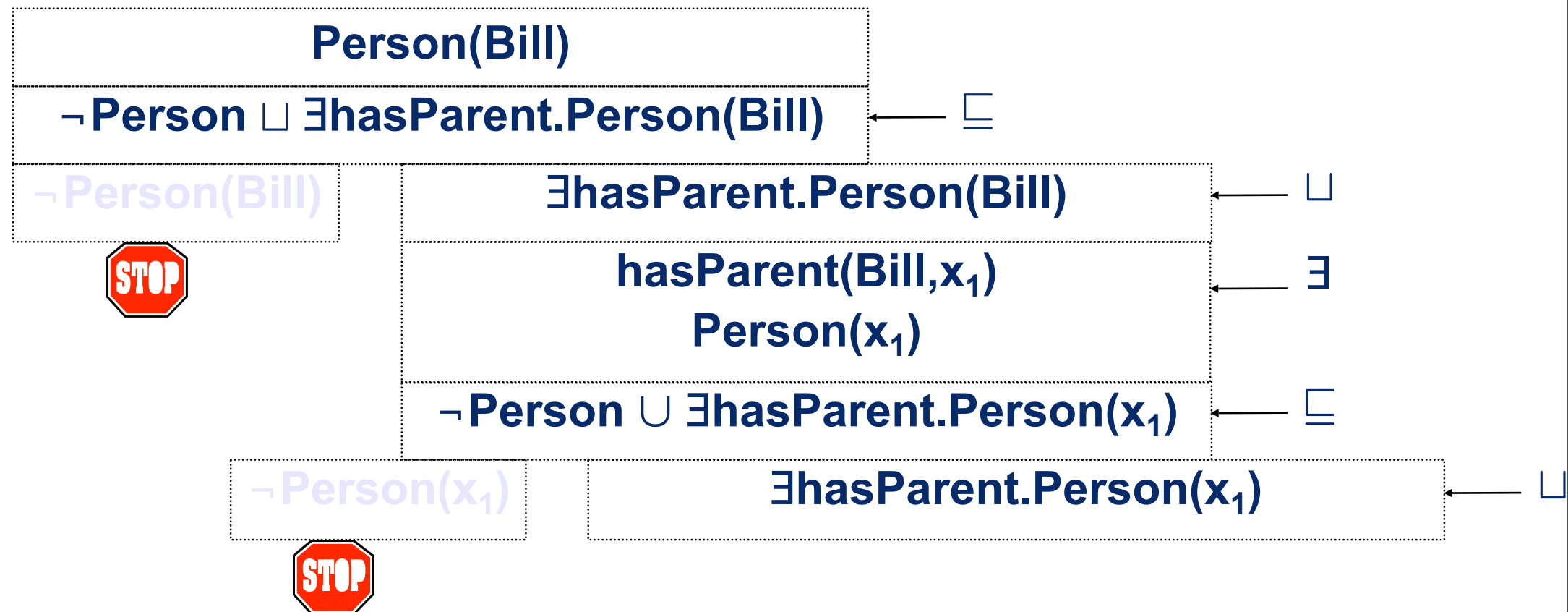
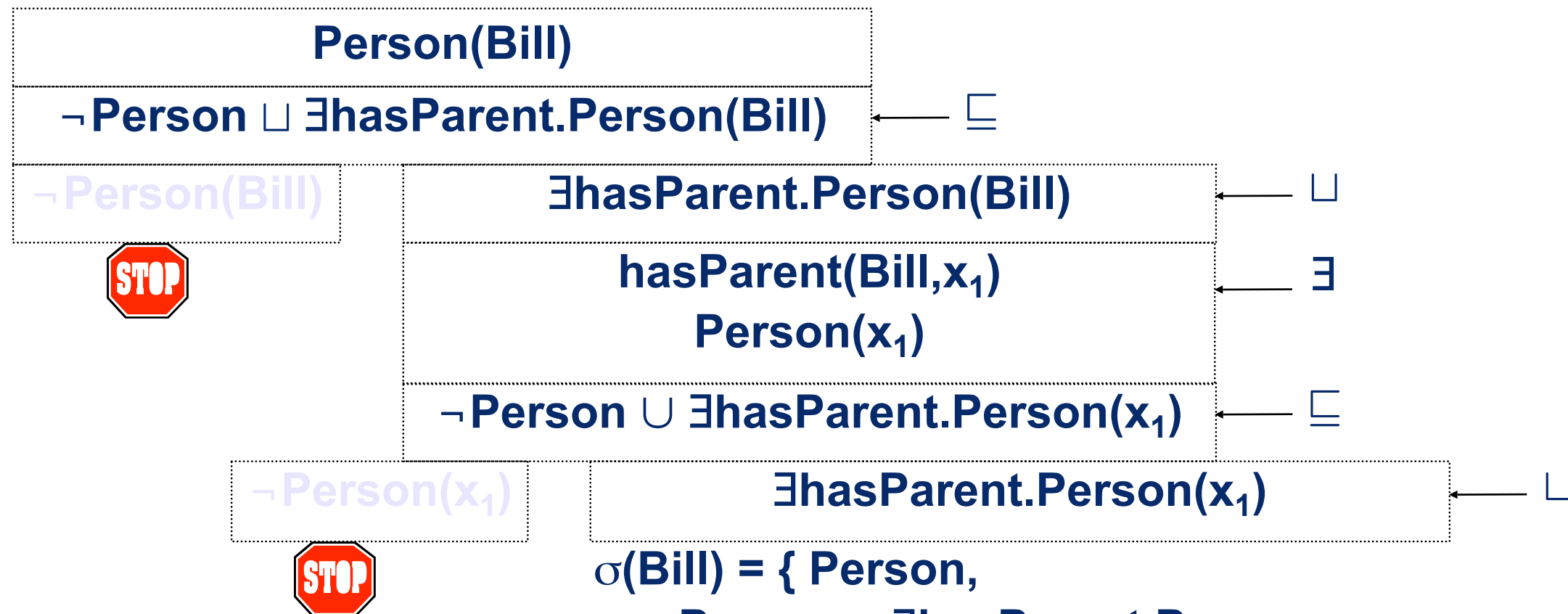


TABLEAU MIT BLOCKING

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$



$\sigma(\text{Bill}) = \{ \text{Person},$
 $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person},$
 $\exists \text{hasParent}.\text{Person} \}$

$\sigma(x_1) = \{ \text{Person},$
 $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person},$
 $\exists \text{hasParent}.\text{Person} \}$

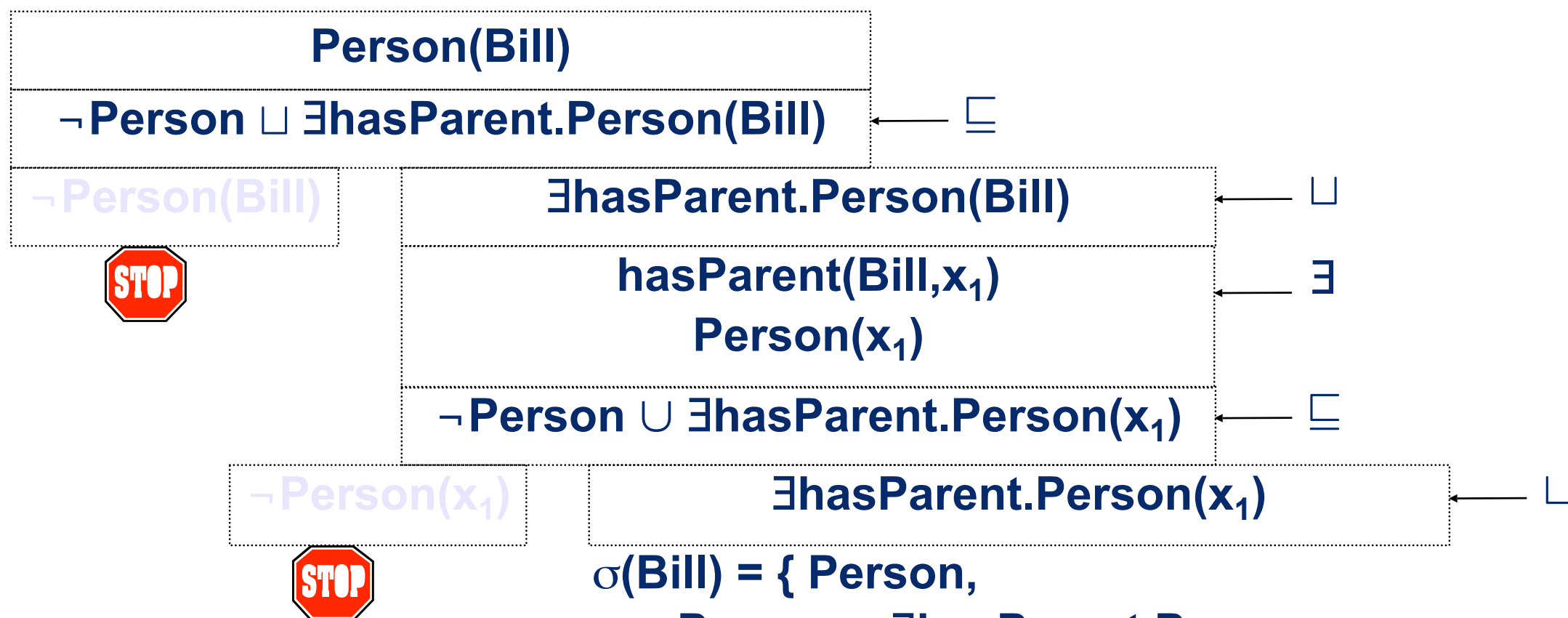
$\sigma(x_1) \sqsubseteq \sigma(\text{Bill})$, so Bill blocks x_1



TABLEAU MIT BLOCKING

- Einziges Axiom: $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person}$

Abzuleiten: $\neg \text{Person}(\text{Bill})$



$\sigma(\text{Bill}) = \{ \text{Person},$
 $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person},$
 $\exists \text{hasParent}.\text{Person} \}$

$\sigma(x_1) = \{ \text{Person},$
 $\neg \text{Person} \sqcup \exists \text{hasParent}.\text{Person},$
 $\exists \text{hasParent}.\text{Person} \}$

$\sigma(x_1) \subseteq \sigma(\text{Bill})$, so Bill blocks x_1

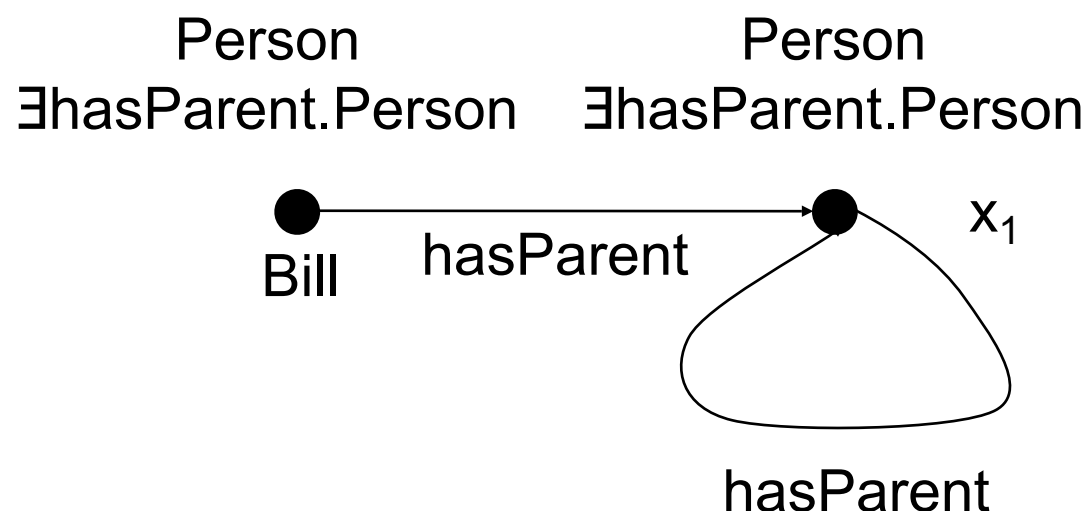


TABLEAU - BLOCKING - DEFINITION



Die Auswahl von $(\exists R.C)(a)$ im Tableauxzweig A ist *blockiert*, falls es ein Individuum b gibt, so dass $\{C \mid C(a) \in A\} \subseteq \{C \mid C(b) \in A\}$ ist.

Zwei Möglichkeiten der Terminierung:

1. Abschluss des Tableaus.
Dann Wissensbasis unerfüllbar.
2. Keine ungeblockte Auswahl führt zu Erweiterung.
Dann Wissensbasis erfüllbar.

TABLEAU FÜR OWL DL



- Die Grundidee ist dieselbe!
- Kompliziertere Blockingregeln müssen verwendet werden.
- Schlechte Unterstützung von Instanzgenerierung.
- Tableau mit Blocking ist $2NExptime$!
→ schlechter als nötig!

TABLEAU-BEWEISER

AIFB 

- Fact
 - <http://www.cs.man.ac.uk/~horrocks/FaCT/>
 - SHIQ
- Fact++
 - <http://owl.man.ac.uk/factplusplus/>
 - SHOIQ(D)
- Pellet
 - <http://www.mindswap.org/2003/pellet/index.shtml>
 - SHOIN(D)
- RacerPro
 - <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>
 - SHIQ(D)

OWL 2 – Syntax und Semantik

Sebastian Rudolph

Institut AIFB · Universität Karlsruhe

Grundlagen Semantic Web (WS09/10)

Seminar für Computerlinguistik, Universität Heidelberg

<http://semantic-web-grundlagen.de>

Foliensatz von M. Krötzsch. Die nichtkommerzielle Vervielfältigung, Verbreitung und Bearbeitung dieser Folien ist zulässig (→ Lizenzbestimmungen CC-BY-NC).

- 1 Einleitung und Ausblick
- 2 XML und URIs
- 3 Einführung in RDF
- 4 RDF Schema
- 5 Logik – Grundlagen
- 6 Semantik von RDF(S)
- 7 OWL – Syntax und Intuition
- 8 OWL – Semantik und Reasoning
- 9 OWL 2 – Syntax und Semantik
- 10 SPARQL – Syntax und Intuition
- 11 Semantik von SPARQL
- 12 Konjunktive Anfragen und Regelsprachen

OWL 2 als „nächste Version“ von OWL

Erweiterungen aufgrund von Praxiserfahrung mit OWL 1.0:

- zusätzliche Ausdruckstärke durch neue ontologische Axiome
- nicht-logische Erweiterungen (Syntax, Metadaten, ...)
- Überarbeitung der OWL-Varianten (Lite/DL/Full)

Zielstellungen:

- weitestgehende Kompatibilität zum existierenden OWL-Standard
- Erhaltung der Entscheidbarkeit von OWL DL
- Behebung von Problemen im OWL-1.0-Standard

Von *SHOIN* zu *SROIQ*

OWL DL basiert auf Beschreibungslogik *SHOIN*(*D*):

- Axiome:
 - Tbox: Subklassenbeziehungen $C \sqsubseteq D$
 - Rbox: Subrollenbeziehungen $R \sqsubseteq S$ (\mathcal{H}), Inverse Rollen R^- (\mathcal{I}), Transitivität
 - Abox: Fakten zu Klassen $C(a)$, Rollen $R(a, b)$, und Gleichheit $a \approx b$ bzw. $a \neq b$
- Klassenkonstruktoren:
 - Konjunktion $C \sqcap D$, Disjunktion $C \sqcup D$, Negation $\neg C$ von Klassen
 - Rollenrestriktionen: universell $\forall R.C$ und existenziell $\exists R.C$
 - Zahlenrestriktionen (\mathcal{N}): $\leq n R$ und $\geq n R$ (n nicht-negative Zahl)
 - Nominale (\mathcal{O}): $\{a\}$
- Datentypen (*D*)

Erweiterung in OWL 2 zu *SROIQ*(*D*)

SHOIN unterstützt verschiedene Abox-Fakten:

- Klassenzugehörigkeit $C(a)$ (C komplexe Klasse),
- Sonderfall: negierte Klassenzugehörigkeit $\neg C(a)$ (C komplexe Klasse),
- Gleichheit $a \approx b$,
- Ungleichheit $a \not\approx b$
- Rollenbeziehungen $R(a, b)$
- *negierte Rollenbeziehungen?*

\rightsquigarrow *SROIQ* erlaubt auch **negierte Rollen** in der Abox: $\neg R(a, b)$

SHOIN unterstützt nur einfache Zahlenrestriktionen (\mathcal{N}):

$\text{Person} \sqcap \geq 3 \text{ hatKind}$

„Klasse aller Personen mit 3 oder mehr Kindern.“

\rightsquigarrow *SROIQ* erlaubt auch **qualifizierte Zahlenrestriktionen** (\mathcal{Q}):

$\text{Person} \sqcap \geq 3 \text{ hatKind.}(\text{Frau} \sqcap \text{Professor})$

„Klasse aller Personen mit 3 oder mehr Töchtern, die Professoren sind.“

Modellierungsaufgabe: „Jeder Mensch kennt sich selbst.“

- *SHOIN*:

kennt(tom, tom) kennt(tina, tina) kennt(udo, udo) ...

↪ nicht allgemein anwendbar

- *SROIQ*: spezieller Ausdruck **Self**

Mensch $\sqsubseteq \exists \text{kennt}.\text{Self}$

SROIQ bietet zusätzliche Aussagen über Rollen:

- $\text{Tra}(R)$: R ist **transitiv** (definiert wie in *SHOIN*)
Beispiel: $\text{Tra}(\text{liegtIn})$
- $\text{Sym}(R)$: R ist **symmetrisch** (definiert wie in *SHOIN*)
Beispiel: $\text{Sym}(\text{verwandtMit})$
- $\text{Ref}(R)$: R ist **reflexiv**, $(x, x) \in R^{\mathcal{I}}$ für alle Domänenindividuen x
Beispiel: $\text{Ref}(\text{kennt})$
- $\text{Irr}(R)$: R ist **irreflexiv**, $(x, x) \notin R^{\mathcal{I}}$ für alle Domänenindividuen x
Beispiel: $\text{Irr}(\text{hatKind})$
- $\text{Dis}(R, S)$: R und S sind **disjunkt**, $(x, y) \notin R^{\mathcal{I}} \cap S^{\mathcal{I}}$ für alle x, y
Beispiel: $\text{Dis}(\text{hatVater}, \text{hatSohn})$
- **Universelle Rolle** U : $(x, y) \in U^{\mathcal{I}}$ für alle x, y
Beispiel: $\top \sqsubseteq \leq 7000000000 U.\text{Menschen}$ (nicht empfohlen!)
 $\rightsquigarrow U$ ist vor allem als Gegenstück zu \top sinnvoll, z.B. als Wurzel der Rollenhierarchie in grafischen Editoren

Allgemeine Rolleninklusion

„Die Freunde meiner Freunde sind auch meine Freunde.“

\rightsquigarrow Kann in *SHOIN* ausgedrückt werden: `hatFreund` ist transitiv.

„Die Feinde meiner Freunde sind auch meine Feinde.“

\rightsquigarrow Kann nicht in *SHOIN* ausgedrückt werden!

Rolleninklusion

- Rbox-Ausdrücke der Form $R_1 \circ R_2 \circ \dots \circ R_n \sqsubseteq S$,
Beispiel: `hatFreund` \circ `hatFeind` \sqsubseteq `hatFeind`
- Semantik: wenn $(x_0, x_1) \in R_1^I, (x_1, x_2) \in R_2^I, \dots, (x_{n-1}, x_n) \in R_n^I$,
dann gilt auch $(x_0, x_n) \in S^I$
Beispiel: wenn $(x, y) \in \text{hatFreund}^I$ und $(y, z) \in \text{hatFeind}^I$,
dann gilt auch $(x, z) \in \text{hatFeind}^I$

Weitere Beispiele:

`teilVon` \circ `gehört` \sqsubseteq `gehört`

`hatBruder` \circ `hatKind` \sqsubseteq `istOnkelVon`

Ausdrucksstärke der Rolleninklusion

Wie kompliziert ist Rolleninklusion?

Mit Rboxen kann man formale Sprachen kodieren: (Skizze!)

Grammatik für Sprache der Wörter $ab, aabb, aaabbb, \dots$:

$$\begin{array}{ll} L ::= ab & \\ L ::= aLb & \end{array} \quad \text{wird zu Rbox} \quad \begin{array}{l} R_a \circ R_b \sqsubseteq L \\ R_a \circ L \circ R_b \sqsubseteq L \end{array}$$

- $\rightsquigarrow \exists L. \top \neq \perp$ (“ $\exists L. \top$ notwendig nicht-leer”) bedeutet*:
„Es gibt eine Kette aus R_a und R_b , die zur Sprache gehört.“
- $\rightsquigarrow \exists L_1. \exists L_2. \top \neq \perp$ für zwei kodierte Sprachen L_1 und L_2 bedeutet:
„Es gibt ein Wort, das zu L_1 und zu L_2 gehört.“

*) bei entsprechender Tbox!

Leider gilt: Leerheit der Überschneidung kontextfreier Sprachen ist unentscheidbar.

\rightsquigarrow OWL mit Rolleninklusionen ist unentscheidbar

Reguläre Rboxen

Kann man Rolleninklusion zwecks Entscheidbarkeit einschränken?

- Rboxen sind wie Grammatiken für kontextfreie formale Sprachen
- Überschneidungen von kontextfreien Sprachen problematisch

⇒ Einschränkung auf reguläre Sprachen!

Reguläre Rboxen

Rollenamen werden mit \prec geordnet (strenge totale Ordnung).
Jede Rbox-Inklusion muss eine der folgenden Formen haben:

- $R \circ R \sqsubseteq R$
- $R \circ S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$
- $R^- \sqsubseteq R$
- $S_1 \circ S_2 \circ \dots \circ S_n \circ R \sqsubseteq R$
- $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$

Dabei gilt: $S_i \prec R$ für alle $i = 1, 2, \dots, n$.

Rbox ist regulär, wenn es so eine Ordnung \prec gibt.

Reguläre Rboxen – Beispiel

Beispiel:

$$R \circ S \sqsubseteq R \quad S \circ S \sqsubseteq S \quad R \circ S \circ R \sqsubseteq T$$

\rightsquigarrow ist regulär mit Ordnung $S \prec R \prec T$

Beispiel:

$$R \circ T \circ S \sqsubseteq T$$

\rightsquigarrow ist nicht regulär (unzulässige Inklusions-Form)

Beispiel:

$$R \circ S \sqsubseteq S \quad S \circ R \sqsubseteq R$$

\rightsquigarrow ist nicht regulär (keine gültige Ordnung möglich)

Beschränkung einfacher Rollen

- Einfache Rollen in *SHOIN* = Rollen ohne transitive Unterrollen
- In *SROIQ*: Beachtung der Rolleninklusionen nötig!

Einfache Rollen sind alle Rollen ...

- die nicht auf der rechten Seite einer Rolleninklusion vorkommen,
- die Inverse von anderen einfachen Rollen sind,
- die nur auf der rechten Seite von Rolleninklusionen vorkommen, bei denen links ausschließlich einfache Rollen stehen.

(Achtung: induktive Definition)

↪ nicht-einfach sind Rollen, die direkt oder indirekt von sich selbst abhängen (und deren Überrollen)

Warum ist das wichtig?

Ausdrücke $\leq n R.C$, $\geq n R.C$, $\text{Irr}(R)$, $\text{Dis}(R, S)$, $\exists R.\text{Self}$ $\neg R(a, b)$
nur für einfache Rollen R und S erlaubt!

(Grund: Sicherstellung von Entscheidbarkeit)

Klassenausdrücke

Klassennamen	A, B
Konjunktion	$C \sqcap D$
Disjunktion	$C \sqcup D$
Negation	$\neg C$
Exist. Rollenrestr.	$\exists R.C$
Univ. Rollenrestr.	$\forall R.C$
Self	$\exists S.\text{Self}$
Größer-als	$\geq n S.C$
Kleiner-als	$\leq n S.C$
Nominale	$\{a\}$

Rollen

Rollennamen	R, S, T
einfache Rollen	S, T
Inverse Rollen	R^{-}
Universelle Rolle	U

Tbox (Klassenaxiome)

Inklusion	$C \sqsubseteq D$
Äquivalenz	$C \equiv D$

Rbox (Rollenaxiome)

Inklusion	$R_1 \sqsubseteq R_2$
Allgemeine Inkl.	$R_1^{(-)} \circ \dots \circ R_n^{(-)} \sqsubseteq R$
Transitivität	$\text{Tra}(R)$
Symmetrie	$\text{Sym}(R)$
Reflexivität	$\text{Ref}(R)$
Irreflexivität	$\text{Irr}(S)$
Disjunktheit	$\text{Dis}(S, T)$

Abox (Fakten)

Klassenzugehörigkeit	$C(a)$
Rollenbeziehung	$R(a, b)$
Neg. Rollenbeziehung	$\neg S(a, b)$
Gleichheit	$a \approx b$
Ungleichheit	$a \not\approx b$

Wie kompliziert ist *SROIQ*?

Rückblick: *SHOIN* (OWL DL) ist sehr komplex (NEXPTIME)

Wie komplex ist *SROIQ*?

Beobachtung: einige Ausdrucksmittel sind nicht wirklich nötig!

- $\text{Tra}(R)$ durch $R \circ R \sqsubseteq R$ ausdrückbar
- $\text{Sym}(R)$ durch $R^- \sqsubseteq R$ ausdrückbar
- $\text{Irr}(S)$ durch $\top \sqsubseteq \neg \exists S.\text{Self}$ ausdrückbar
- Universelle Rolle durch transitive, reflexive Überrolle aller Rollen ersetzbar (hier nicht vertieft)
- Abox durch Nominale darstellbar, z.B. $R(a, b)$ durch $\{a\} \sqsubseteq \exists R.\{b\}$

Qualifizierte Zahlenrestriktionen kaum problematisch (bekannt und implementiert, siehe Vorlesung zu OWL)

~> Hauptproblem Rollenaxiome (Rbox)

Wie geht man mit Rboxen um?

- Rbox-Regeln ähneln formalen Grammatiken
- jede Rolle R definiert eine reguläre Sprache:
die Sprache der Rollen-Ketten, aus denen R folgt
- reguläre Sprachen \equiv reguläre Ausdrücke \equiv endliche Automaten

\rightsquigarrow Ansatz: Tableauverfahren werden mit „Rbox-Automaten“ erweitert

Details siehe Literaturangaben zu SROIQ

Tableauverfahren von *SROIQ* zeigt:

SROIQ ist entscheidbar.

- Algorithmus hat gute Anpassungseigenschaften: ungenutzte Merkmale belasten die Abarbeitung kaum („pay as you go“)
- aber: Tableau-Verfahren ungeeignet für enge Komplexitätsabschätzungen
- besser: Rückführung auf andere Logiken mit bekannter Komplexität \rightsquigarrow Komplexität von *SROIQ*: $N2EXPTIME$

*SR**OIQ* ist „nur“ logische Grundlage von OWL 2 DL

Weitere nicht-logische Aspekte:

- Syntax (Erweiterung nötig)
- Datentypdeklaration und Datentypfunktionen, neue Datentypen?
- Metamodellierung: „Punning“
- Kommentarfunktionen und ontologische Metadaten
- Invers-funktionale konkrete Rollen (DatatypeProperties): Keys?
- Mechanismen zu Ontologieimport?
- ...

Metamodellierung

Spezifikation ontologischen Wissens *über* einzelne Elemente der Ontologie (einschließlich Klassen, Rollen, Axiome).

Beispiele:

- „Die Klasse *Person* wurde am 30.1.2008 von *MarkusK* angelegt.“
- „Für die Klasse *Stadt* wird die Property *Einwohnerzahl* empfohlen.“
- „Die Aussage ‚Dresden wurde 1206 gegründet‘ wurde maschinell ermittelt mit einer Sicherheit von 85%.“

Wortspiele in OWL: Punning

Metamodellierung in ausdrucksstarken Logiken ist gefährlich und teuer!

OWL 2. unterstützt zurzeit einfachste Form von Metamodellierung:

Punning

- Bezeichner für Klassen, Rollen, Individuen müssen nicht disjunkt sein
- keine *logische* Beziehung zwischen Klasse, Individuum und Rolle gleichen Namens
- Beziehung nur relevant für pragmatische Interpretation

Beispiel:

Person(Sebastian) klasseErstelltVon(Person, Markus)

Punning unterstützt einfache Metadaten mit (schwacher) semantischer Bedeutung

Wie kann man rein syntaktische Kommentare zu einer Ontologie machen?

- Kommentare in XML-Dateien: `<!-- Kommentar -->`
 \rightsquigarrow kein Bezug auf OWL-Axiome dieser Datei
- nicht-logische Annotationen in OWL:
 `owl:AnnotationProperty`
 \rightsquigarrow fest verknüpft mit (semantischem) ontologischem Element, kein syntaktischer Bezug

OWL 2 soll „echte“ syntaktische Kommentare unterstützen

zwei grundlegende Varianten:

- Funktionale Syntax: ersetzt „Abstrakte Syntax“ von OWL 1.0
- RDF-Syntax: Erweiterung von OWL/RDF 1.0

⇒ funktionale Syntax einfacher zu definieren
(keine RDF-Beschränkungen), kompakter

⇒ RDF-Syntax für Abwärtskompatibilität wichtig

Quo vadis, OWL Lite?

OWL Lite als Fehlschlag:

- beinahe so komplex wie OWL DL
- komplizierte Syntax gibt keinen direkten Zugang zu wahrer Ausdrucksstärke
- Verwendung in Ontologien heute praktisch nur „zufällig“, nicht bewusst

Ursprüngliches Ziel:

einfach und effizient implementierbarer Teil von OWL

⇒ 3 *tractable profiles* für OWL 2: EL, RL und QL

basiert auf Beschreibungslogik \mathcal{EL}^{++}

- Konzipiert nur mit Konjunktion $C \sqcap D$, Existenz $\exists R.C$, \top und \perp
- Nominale
- allgemeine Rolleninklusionen (Rbox), Transitivität

Vorteile:

- polynomielle Komplexität
- schnelle Implementierungen verfügbar
- einfache Definition
- unterstützt praktisch relevante Ontologien (SNOMED)

basiert auf Beschreibungslogik DLP

- in Subklassen-Axiomen sind auf rechter und linker Seite unterschiedliche Arten Klassen erlaubt
- Inverse Rollen, RIAs, einfache Rollenhierarchien
- Abox wie in *SROIQ*

Vorteile:

- polynomielle Komplexität (verwandt mit Logikprogrammierung)
- schnelle Implementierungen verfügbar (Oracle)
- unterstützt RDFS

basiert auf Beschreibungslogik DL Lite_R

- Konzipiert nur mit Konjunktion $C \sqcap D$, unqualifizierter Existenz $\exists R.T$, und \perp
- Inverse Rollen, einfache Rollenhierarchien
- Abox wie in *SROIQ*

Vorteile:

- sub-polynomielle Komplexität (verwandt mit relationalen Datenbanken)
- schnelle Implementierungen verfügbar
- relativ einfache Definition

OWL 2 als erste Weiterentwicklung des OWL-Standards

- logische Erweiterung: Beschreibungslogik *SR_QIQ* als Grundlage
- neue Ausdrucksmittel vor allem Rollenaxiome, qualifizierte Zahlenrestriktionen
- nicht-logische Erweiterungen: Punning, Kommentare, Datentypen, u.a.
- Einführung von 3 neuen OWL profiles mit polynomieller Komplexität
- Verabschiedung Oktober 2009